Computer Science Department

TECHNICAL REPORT

A LAGRANGIAN FRACTIONAL STEP METHOD FOR THE INCOMPRESSIBLE NAVIER-STOKES EQUATIONS

By

Christoph Börgers

October 1985

Technical Report #183

NEW YORK UNIVERSITY

Department of Computer Science
Courant Institute of Mathematical Sciences
251 MERCER STREET, NEW YORK, N.Y. 10012

A LAGRANGIAN FRACTIONAL STEP METHOD FOR THE INCOMPRESSIBLE NAVIER-STOKES EQUATIONS

By

Christoph Börgers

October 1985

Technical Report #183

Contents

Title: A Lagrangian Fractional Step Method for the Incompressible Navier-Stokes Equations

Author: Christoph Börgers

Advisor: Charles S. Peskin

Abstract: We develop a modification of Peskin's Lagrangian fractional step method for the incompressible Navier-Stokes equations. This new method is substantially more efficient than the one originally proposed by Peskin. On a grid with N points, the work per time step is proportional to NlogN. This gain in efficiency is accomplished by modifying the splitting and by using a multigrid method for the solution of the resulting systems of equations.

The method uses finite difference operators constructed with the aid of Voronoi diagrams. We have implemented it on a periodic domain in the plane. We describe an efficient algorithm for the numerical construction of periodic Voronoi diagrams in the plane. We believe that this algorithm can easily be generalized to higher space dimensions.

We report on numerical experiments with our method. We solve test problems with known solutions, and we compute a flow evolving from an initial vortex blob. Our results indicate that the method is convergent of first order.

As an application of our method, we present a fully Lagrangian variant of Peskin's algorithm for the treatment of elastic boundaries immersed in the fluid, and we report on numerical results obtained with this algorithm.

Acknowledgements

I am deeply grateful to Professor Charles S. Peskin for his instructive, enjoyable and encouraging way of working with me, without ever becoming impatient at my weaknesses.

I am equally grateful to Professor Olof B. Widlund for three years of invaluable help and advice. I thank him for all I have learned from him, for all the time and consideration which he has devoted to my education, and for his generous kindness.

Studying at the Courant Institute has been a great and rewarding experience. I am thankful for all I have been taught, for the friendly and generous atmosphere at the institute, and for financial support from the Courant Institute and the Graduate School of Arts and Science at New York University.

I thank the mathematics group at the Lawrence Berkeley Laboratory for the interesting and pleasant summer which I spent there as a visitor in 1984.

1. Introduction

Most of the methods presently used in computational fluid dynamics use the Eulerian rather than the Lagrangian coordinate system. The strength of Lagrangian methods for the Navier-Stokes equations is the natural and simple way in which the non-linear convection part of the equations is treated, by moving the grid in the flow. However, this movement of the grid causes crucial problems. The discretization of the differential operators and the numeric solution of the resulting discrete problems are more difficult on irregular grids than on the regular grids used for Eulerian computations.

These difficulties have hampered the development of fully Lagrangian methods for flow with large deformations in two or three dimensions. Methods which combine Lagrangian and Eulerian features have been studied for some time. Examples are the MAC (Marker and Cell) method due to Harlow and Welch (1965), in which Lagrangian markers are used to determine the position of a free boundary, but not for the solution of the Navier-Stokes equations, and the ALE (Arbitrary Lagrangian-Eulerian) method, in which rezoning steps prevent the mesh from changing its initial topology; see Hirt, Amsden and Cook (1974), Chan (1975).

There has been recent progress in overcoming the difficulties inherent in the use of fully Lagrangian methods. Discretizations of differential operators on irregular grids can be computed with increased efficiency by using new computational geometry algorithms. The solution of the resulting discrete problems has become easier due to new numerical techniques such as multigrid methods. We briefly discuss these two developments in turn.

In order to discretize a differential operator on a general set S of points in the plane, one usually needs a mesh associated with S, for example a set of polygons each containing exactly one of the points in S, or a triangulation whose vertices are the points

in S. The Voronoi diagram associated with S can serve as a polygonal mesh or as a tool for the construction of a triangular mesh. The Voronoi polygon associated with $X \in S$ is the set of points at least as close to X as to any other point in S; see Voronoi (1908). Recently many algorithms for the construction of Voronoi diagrams have been developed; see Shamos and Hoey (1975) for a very fast sequential algorithm, and Aggarwal, Chazelle, Guibas, O'Dúnlaing and Yap (1985) for a parallel algorithm. Peskin (1979) introduced an algorithm for updating Voronoi diagrams within a time dependent calculation. For N points, O(N) operations are required per time step. This algorithm can be generalized to three dimensions with few changes. We note that Voronoi diagrams have been used in Lagrangian fluid dynamics codes by Augenbaum (1982), Peskin (1979), Trease (1981).

In addition, efficient iterative reconnection algorithms for triangular meshes have been introduced; see Crowley (1971), Dukowicz (1981), Fritts and Boris (1979). The triangulations thus generated are closely related to Voronoi diagrams and can in fact be used to construct Voronoi diagrams. The generalization of these algorithms to three dimensions is possible, but complicated; see Fritts (1985).

A further mesh construction algorithm applicable to Lagrangian fluid dynamics has recently been introduced by Boris (1985). Given $N = N_1 \cdot N_2$ arbitrary points in the plane, this algorithm finds an ordering $X(i_1,i_2)$, $1 \le i_1 \le N_1$, $1 \le i_2 \le N_2$ such that

$$X_1(i_1,i_2) \le X_1(i_l+1,i_2), \quad X_2(i_1,i_2) \le X_2(i_l,i_2+1) \text{ for all } i_1,i_2.$$

This indexing is called an MLG (Monotonic Logical Grid). It is easy to see that an MLG always exists and can be computed in O(NlogN) operations. Boris describes a simple iterative procedure by which it can be updated within a time dependent computation in only O(N) operations per time step. The algorithm remains unchanged in arbitrary dimensions and is easily vectorizable.

We turn to the numerical solution of equations on Lagrangian grids. Many numerical algorithms for incompressible fluid dynamics require the solution of Poisson

equations for the pressure. An example is the projection method introduced by Chorin (1968). We shall consider a Lagrangian version of the projection method which requires the solution of a Poisson equation for the pressure and a Helmholtz equation for each velocity component. We use a multigrid algorithm, more precisely a twogrid algorithm, which works well even on random grids. A different successful multigrid algorithm for discrete Poisson equations on irregular grids has been introduced by Löhner, Morgan, Peraire and Zienkiewicz (1985). The new AMG (Algebraic Multigrid) algorithms appear to be still another possibility; see, e.g., Stüben (1983). They have apparently not yet been used in Lagrangian fluid dynamics.

Most of the recent work on Lagrangian methods concentrates on inviscid flow. In this thesis, however, we shall study a Lagrangian method for the viscous, incompressible Navier-Stokes equations, using primitive variables. We now give an introduction to this method.

In 1979, Peskin introduced a new Lagrangian method for the incompressible Navier-Stokes equations, using finite difference operators constructed with the aid of Voronoi diagrams. The method requires the solution of a linear symmetric, indefinite system of equations, essentially a discretization of the Stokes equations, in every time step. No efficient solver was found for this problem. Only few numerical tests were therefore performed with the method, all using very small grids.

In this thesis, a fractional step method which is a modification of Peskin's original method is developed. Only discrete Helmholtz and Poisson problems need to be solved in every time step. The much improved efficiency due to the two-level iteration has allowed us to conduct experiments on grids of substantially larger size. The present method cannot yet compete with existing Eulerian methods for the incompressible Navier-Stokes equations. We believe, however, that this work shows that the method is promising.

We give a short summary. The two-dimensional incompressible Navier-Stokes equations have the form

$$\underline{u}_t + (\underline{u}\cdot\underline{\nabla})\underline{u} - \nu\Delta\underline{u} + \underline{\nabla}p = \underline{f} \qquad (1.1)$$
$$\underline{\nabla}\cdot\underline{u} = 0\ ,$$

where $\underline{u}=\underline{u}(\underline{x},t)\in R^2$, $p=p(\underline{x},t)\in R$, $\underline{x}\in R^2, t\in R$, $t>0$. We impose an initial condition

$$\underline{u}(\underline{x},0)=\underline{u}_0(\underline{x}) \qquad (1.2)$$

and the periodicity conditions

$$\underline{u}(\underline{x}+\underline{k},t)=\underline{u}(\underline{x},t), \quad p(\underline{x}+\underline{k},t)=p(\underline{x},t), \quad \underline{k}\in Z^2. \qquad (1.3)$$

In section 2, we describe a way of adapting Peskin's 1979 algorithm for the construction of Voronoi diagrams to the case of a periodic domain. Peskin also introduced discretizations of the Laplace, gradient and divergence operators on irregular grids in the plane, based on the Voronoi diagram; see Peskin (1979). We present the derivation and properties of these operators in section 3. In section 4, we describe the two-level iteration and give results of numerical experiments which indicate that the method is almost as efficient as multigrid algorithms for Helmholtz equations on regular grids. In section 5, we consider the problem of projecting a vector field onto its divergence free component. We describe an iterative algorithm which is applicable for irregular grids and which quickly finds a good approximation to the solution of the continuous problem. Convergence to the solution of the discrete projection problem is provable but slow. In section 6, we combine these tools to obtain a fractional step method for the Navier-Stokes equations. Numerical experiments suggest that the method is of first order in the sense that the discretization error is roughly reduced by 1/2 if the number N of fluid markers is multiplied by 4 and the time step is reduced by 1/2. (In two space dimensions, the effective meshwidth h is $O(N^{-1/2})$, therefore multiplication of N by 4 reduces h by 2.) The work per time step is O(NlogN) and could be reduced to O(N) by a trivial modification. The method is implicit, and there is no stability condition on the size of the

time step. In section 7, we describe an algorithm for immersed boundaries due to Peskin (1977). We note that (1.3) is an appropriate condition in this context. The immersed boundaries are represented by chains of Lagrangian particles connected by springs. For the fluid, Peskin uses an Eulerian method, which we replace by our Lagrangian method. We present some preliminary numerical results. In section 8, we discuss difficulties with our method and suggest ways of overcoming them.

2. Construction of periodic Voronoi diagrams

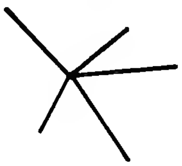
In section 2.1, we outline the algorithm, omitting details of implementation unless they are useful for a general description of the method. In section 2.2, we give algorithmic details and discuss our implementation of the method.

2.1 The algorithm

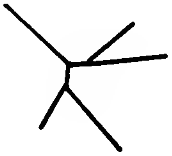
Let S be a set of points in R^2. For $\underline{X} \in S$, we define

$$P(\underline{X},S) := \{\underline{Y} \in R^2 : \|\underline{Y}-\underline{X}\| \leq \|\underline{Y}-\hat{\underline{X}}\| \text{ for all } \hat{\underline{X}} \in S\}. \tag{2.1}$$

$P(\underline{X},S)$ is an intersection of half planes, i.e. a convex polygon. It is called the Voronoi polygon (Voronoi, 1908) or Dirichlet region of $\underline{X}$ with respect to S. The collection of all $P(\underline{X},S)$, $\underline{X} \in S$, is called the Voronoi diagram associated with S and is denoted by $V(S)$. An edge in $V(S)$ is generated by two points, i.e. it belongs to two polygons. A corner is generated by three or more points, i.e. it belongs to three or more polygons. We always assume that every corner is generated by exactly three points. A degenerate case such as



is resolved into



This resolution of multiple corners into sets of simple corners linked by edges of length zero is always possible but never unique. This fact is of some importance for our construction algorithm described below.

It is well known that the Voronoi diagram for a set of N points in the plane can be constructed in O(NlogN) operations and that this operation count is best possible; see

Shamos and Hoey (1975). However, if additional information on S is available, better operation counts can be achieved. The algorithm introduced by Peskin (1979) makes use of the assumption that the Voronoi diagram has already been constructed for a set $\bar{S}$ from which S can be obtained by slightly displacing each point, and that certain pieces of information about this construction have been saved. Similarly, the algorithm used by Fritts (1981) can be viewed as an algorithm which uses V($\bar{S}$) to compute V(S). Experiments indicate that both algorithms construct V(S) in linear time. If S is a random set, there may be algorithms for which the expected value of operations is smaller than O(NlogN). This is particularly plausible if some information is given about the probability distribution of S. Numerical experiments supporting this statement will be presented later in this section.

We now consider the case

$$S = S_N := \{\underline{X}_j + \underline{k} : 1 < j < N,\ \underline{k} \in Z^2\},\ \text{with}\ \underline{X}_j \in [0,1)^2. \tag{2.2}$$

In our fractional step method, the $\underline{X}_j$ will be fluid markers. The corresponding Voronoi diagram can be viewed as a periodic Voronoi diagram in the plane or as a Voronoi diagram on the torus

$$T := R^2/Z^2 \tag{2.3}$$

with the metric

$$d(\underline{X},\underline{Y}) := \min\{\ \|\underline{X} - \underline{Y} + \underline{k}\| : \underline{k} \in Z^2\ \}. \tag{2.4}$$

We describe a modification of the algorithm due to Peskin (1979) for the construction of $V(S_N)$.

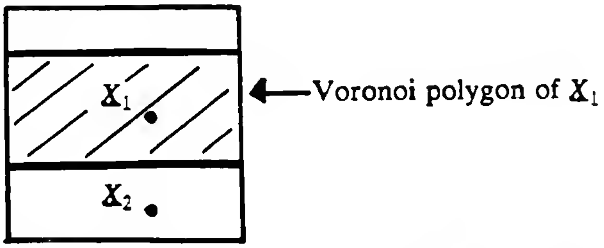
We store information about the corners in $V(S_N)$ which lie in $[0;1)^2$.

Proposition 2.1: $V(S_N)$ has exactly M=2N corners in $[0;1)^2$.

Proof:

One could use the fact that the Euler characteristic of the torus is 0. Notice, however, that the Voronoi polygons on the torus are not necessarily homeomorphic to disks.

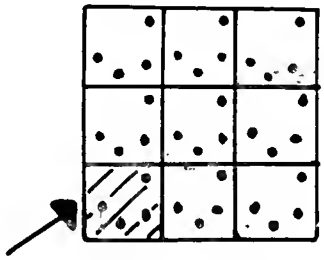
This is shown by the following example, in which the two polygons are homeomorphic to $\{X \in R^2 : 1 \le \|X\| \le 2\}$.



Therefore an additional argument is needed to show M=2N. We note that those polygons on the torus which are not homeomorphic to disks also cause difficulties in our algorithm for the construction of V(S); see below.

Our proof that M=2N is based on the fact that the Euler characteristic of the sphere is 2. Consider m by m copies of the points $X_1, \cdots, X_N$:

(m=3 here)



the unit square

Consider the corresponding Voronoi diagram. Intersect it with $[0; m]^2$. The result is a polygonal mesh. We extend this mesh to one on the sphere $R^2 \cup \{\infty\}$ by introducing edges which connect the four corners of $[0; m]^2$ with ∞. This introduces a degenerate corner at ∞ which we resolve into two non-degenerate corners connected by an edge of length 0. The mesh on the sphere has m^2N+4 faces. Since every edge joins 2 corners and every corner belongs to 3 edges, the number of corners is 2/3 times the number of edges. From Euler's formula, we now conclude that the total number of corners, including those at ∞, is $2m^2N+4$.

The (m-2) by (m-2) inner squares are identical up to translation and contain $(m-2)^2M$ corners. For fixed N, the number of corners in the 4m-4 boundary squares is $O(m)$. Therefore we have

$$(m-2)^2M + O(m) = 2m^2N + 2 .$$

Letting $m \to \infty$, we obtain $M = 2N$. ■

It follows from this that there are exactly 3N edges (modulo Z^2) in $V(S)$. This is of interest for our method because edges in $V(S)$ correspond to nonzero couplings in the finite difference operators presented in section 3.

We call the 2N corners in $[0;1)^2$ the stored corners and call the points $\underline{X}_1, \cdots, \underline{X}_N$ the stored points. For the j-th stored corner, we store its cartesian coordinates (XC(j),YC(j)) and the square RAD2(j) of its radius, which is defined as its distance from its generating points. We introduce the notation $rad(\underline{c})$ for the radius of the corner $\underline{c}$. The cartesian coordinates of a corner in $V(S)$ can be written in the form

$$\mathrm{XC(IC)+KX,\ YC(IC)+KY}$$

with uniquely determined integer indices IC,KX,KY. We call IC the basic index and KX,KY the shift indices. We call the corner with basic index IC and shift indices KX,KY the corner (KX,KY,IC). We use analogous conventions for the points in S, i.e. the fluid markers and their translations. We store integer indices

$\mathrm{IPT}(i,n,j)\ ,\quad i=1,2,3\ ,\quad n=0,1,2$ and
$\mathrm{ICR}(i,n,j)\ ,\quad i=1,2,3\ ,\quad n=0,1,2,$

which have the following meaning. The point

$$(\mathrm{IPT}(i,1,j),\mathrm{IPT}(i,2,j),\mathrm{IPT}(i,0,j))$$

is the i-th generating point of the j-th stored corner, and the corner

$$(\mathrm{ICR}(i,1,j),\mathrm{ICR}(i,2,j),\mathrm{ICR}(i,0,j))$$

is the i-th neighbor corner of the j-th stored corner.

We now outline the algorithm for the construction of $V(S_N)$.

Algorithm 2.1:

Step 1: Construct $V(S_1)$.

Step 2: For j=2,...,N:

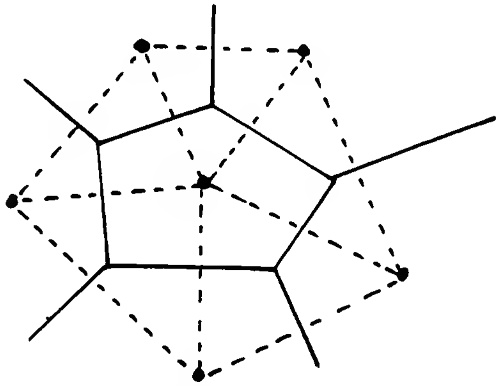
Step 2.1: Construct $V(S_{j-1} \cup \{\underline{X}_j\})$ from $V(S_{j-1})$.

Step 2.2: Construct $V(S_j)$ from $V(S_{j-1} \cup \{\underline{X}_j\})$.

Thus in step 2.1, we add the stored point $\underline{X}_j$ without its periodic images to the otherwise periodic Voronoi diagram. In step 2.2, we then add all the periodic images of $\underline{X}_j$.

Step 1 is trivial. The central part of the algorithm is a procedure for the construction of $V(S \cup \{\underline{Y}\})$ if $\underline{Y} \notin S$ and V(S) is known, i.e. step 2.1. This procedure was described by Peskin (1979) and Bowyer (1981) independently. For completeness, we present it here.

A corner $\underline{c}$ in V(S) is called broken by $\underline{Y}$ if it is not a corner in $V(S \cup \{\underline{Y}\})$. $\underline{Y}$ breaks $\underline{c}$ if and only if $\underline{Y}$ lies in the open disk around $\underline{c}$ with radius $rad(\underline{c})$. $\underline{Y}$ always breaks at least one corner. This is, in general, guaranteed if $\underline{Y}$ lies in the convex hull of S. To see this, notice that the convex hull of S is exactly the union of the Delaunay triangles, i.e. the triangles which are obtained by joining all pairs of points in S with adjacent Voronoi polygons:

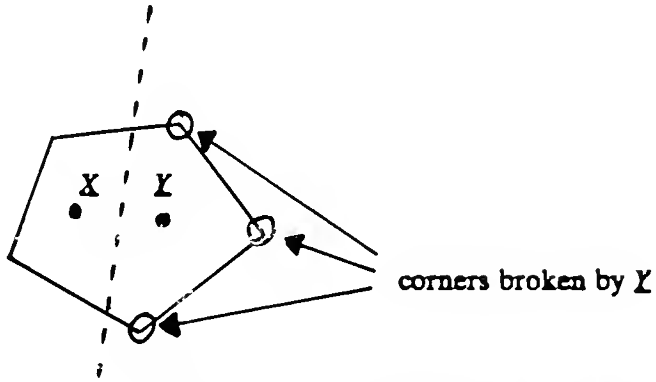


$\underline{Y}$ lies in one of the Delaunay triangles. Consider the circumscribing circle of this Delaunay triangle. Its center is a corner $\underline{c}$ in V(S). Since $\underline{Y}$ lies in the interior of this circle, it breaks $\underline{c}$.

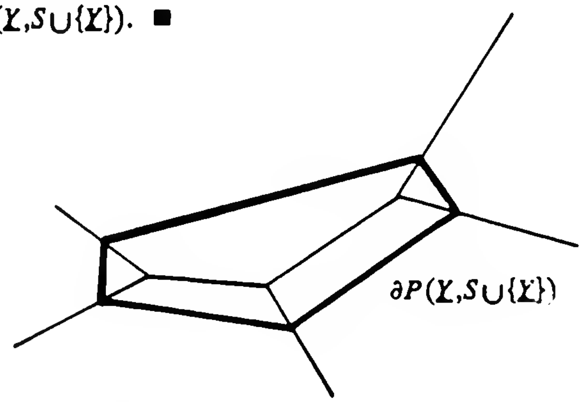
Proposition 2.2: The set of broken corners is connected in V(S).

Proof:

This proof is due to D. Goldfarb (unpublished). The corners of a fixed polygon in V(S) which are broken by $\underline{Y}$ form a connected set:



Walking along the boundary of $P(\underline{Y},S\cup\{\underline{Y}\})$, one visits all polygons which have corners broken by $\underline{Y}$. Whenever $\partial P(\underline{Y},S\cup\{\underline{Y}\})$ crosses an edge of V(S), this edge has exactly one endpoint which is broken by $\underline{Y}$. This endpoint is a common corner of the two polygons bordered by the edge. Therefore, a walk through all broken corners is constructed by following $\partial P(\underline{Y},S\cup\{\underline{Y}\})$. ■



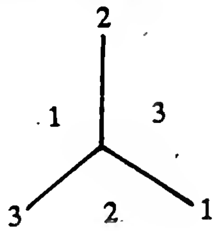
We can therefore find all broken corners in O(1) operations, once we know one of them. We can find a first broken corner in the following way. We first choose some point $\underline{X}\in S$ close to $\underline{Y}$. Good ways of choosing $\underline{X}$ will be discussed below, since the efficiency of the algorithm depends mainly on this choice. We determine a corner $\underline{c}$ of $P(\underline{X},S)$. For this purpose, we use an array IPTCR with the following meaning. The corner

$$(\mathrm{IPTCR}(1,k),\mathrm{IPTCR}(2,k),\mathrm{IPTCR}(0,k))$$

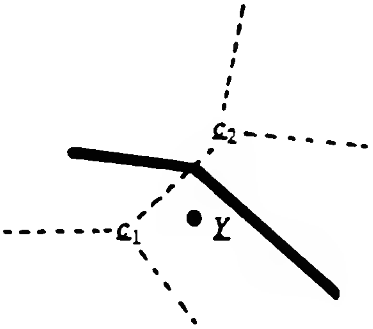
is a corner of the k-th polygon. Notice that the pointers IPT and ICR introduced above

point from corners to points and corners and are therefore different from IPTCR. Beginning with $\underline{c}$, we then conduct a breadth first search through the graph of corners of V(S) until a broken corner is found.

Once we know all corners broken by $\underline{Y}$, we can find all corners of the new polygon $P(\underline{Y},S \cup \{\underline{Y}\}))$ by using the facts that each such corner lies on an edge between a broken and an unbroken corner in V(S) and that on each such edge there is a new corner. It is useful, but nct necessary, to store the neighbor corners of a corner $\underline{c}$ in counterclockwise order and to number the generating points such that the edge connecting $\underline{c}$ with its i-th neighbor corner lies opposite to the i-th generating point:



This ordering is the most important specifically two-dimensional feature of our code. It facilitates the finding of the generating points of a new corner. If the new corner lies on an edge between a broken corner $\underline{c}_1$ and an unbroken corner $\underline{c}_2$ in V(S), and if $\underline{c}_2$ is the j-th neighbor of $\underline{c}_1$ ($j \in \{1,2,3\}$), then the generating points with indices j+1 (mod 3) and j+2 (mod 3) of $\underline{c}_1$ are also generating points of the new corner. The third generating point of the new corner is the newly introduced point $\underline{Y}$:



Notice, however, that these two points could also be obtained by intersecting the sets of generating points of $\underline{c}_1$ and $\underline{c}_2$ in V(S), regardless of the order in which neighbor corners

and generating points are stored.

It remains to describe step 2.2. To simplify the notation, we take j=N and define

$$\hat{P}_N := P(\underline{X}_N, S_{N-1} \cup \{\underline{X}_N\}) \text{ and} \tag{2.5}$$

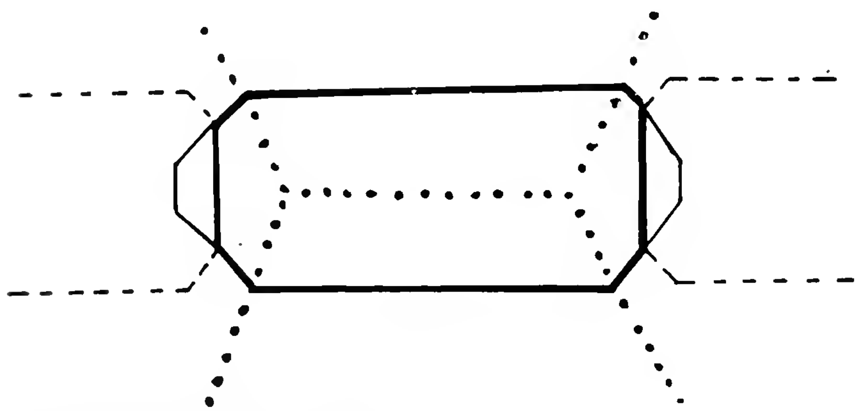
$$P_N := P(\underline{X}_N, S_N). \tag{2.6}$$

It is clear how to get P_N from $\hat{P}_N$:

$$P_N = \hat{P}_N \cap ([X_{N,1} - 0.5; X_{N,1} + 0.5] \times [X_{N,2} - 0.5; X_{N,2} + 0.5]), \tag{2.7}$$

where $X_{N,1}$ and $X_{N,2}$ are the coordinates of $\underline{X}_N$.

We first assume that the decision which corners of $\hat{P}_N$ to cut off can simply be made by using (2.7), by checking the coordinates. It then seems easy to find the radii of the new corners in $V(S_N)$, their neighbor corners and their neighbor points, as indicated in the following figure:

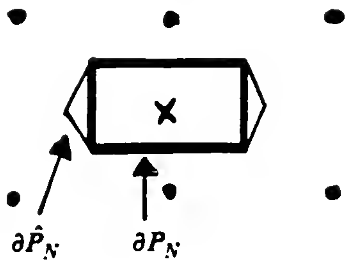


To find the neighbors of the new corners, follow "- - - - -" and "• • • • • •".

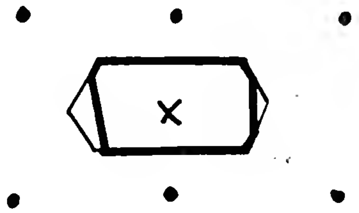
• • • • • : edges in $V(S_{N-1})$; ——— : ∂P_N;

- - - - - : translations of edges of $\hat{P}_N$; ▬▬▬ : $\partial \hat{P}_N$

Therefore the description of step 2.2 seems to be completed. However, while this procedure works correctly in most cases, it occasionally breaks down because of rounding errors. We show a typical situation which leads to a breakdown:

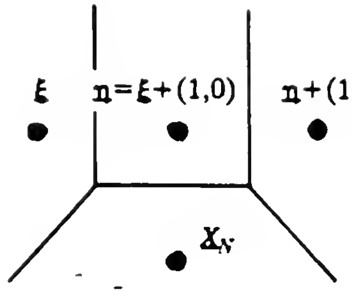


Here N=2, X_1 = (0.0,0.0), X_2 = (0.0,0.5). P_N should have exactly 6 corners, two of which are identical up to translation with two others. In any finitely accurate arithmetic, the computed number of corners of P_N may be 4,5,6,7 or 8. This statement is explained by the following figure, which shows a way of cutting which results in 7 computed corners of P_N.

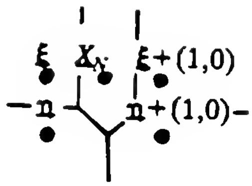


The situations which cause difficulties are cases in which a polygon is adjacent to its own translation. This is most likely to happen at early stages of the construction. It may however even happen in $V(S_N)$ with arbitrarily large N, as when all points $X_1,\cdots,X_N$ lie on a line parallel to one of the coordinate axes. Even if a polygon is adjacent to its own translation, difficulties such as the one illustrated above occur only with a probability which tends to zero with the machine epsilon. It is desirable, though, to have an algorithm which will work, for any given set of points, if the machine epsilon is sufficiently small but positive.

We therefore have to cut $\hat{P}_N$ in a more careful way to obtain P_N. Let c be a corner of $\hat{P}_N$, generated by ξ, η, X_N. Let $\hat{c}$ be a corner of $\hat{P}_N$, generated by $\xi+(1,0)$, $\eta+(1,0)$, X_N. The following figures show two such situations. For simplicity, we use examples in which the points lie on a square grid. In practice, breakdowns do occur even in irregular configurations, even though with small probability.



(a)



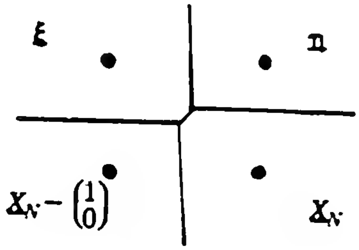
(b)

Case (a) is the one discussed above as an example for a breakdown. A situation such as case (b) occurs, for example, when constructing the Voronoi diagram associated with the set

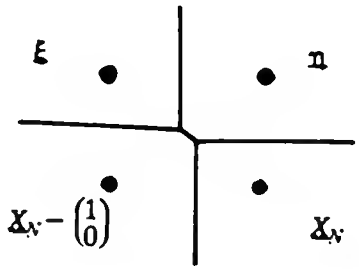
$$\{ X_1=(0,0),\ X_2=(0,\tfrac{1}{2}),\ X_3=(\tfrac{1}{2},\tfrac{1}{2}) \}$$

Here the point being introduced is $X_N = X_3 = (\frac{1}{2},\frac{1}{2})$.

To decide whether $\underline{c}$ is broken by $X_N-(1,0)$ in situation (a), we consider the Voronoi diagram $V(\{\underline{\xi};\ \underline{n};\ X_N;\ X_N - (1,0)\})$:

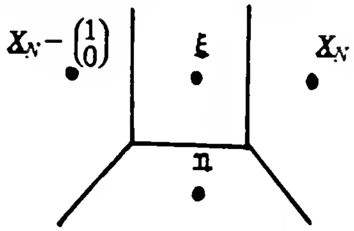


If we draw the diagram like this, we conclude that $\underline{c}$ is not broken by $X_N - (1,0)$, since it is a corner in the four-point Voronoi diagram. Clearly, we can use the same diagram to decide whether $\hat{\underline{c}}$ is broken by $X_N+(1,0)$. We conclude that it is. If we had drawn the short edge of length 0 like this:



we would have concluded that $\underline{c}$ is broken whereas $\hat{\underline{c}}$ is not. In any case, we make opposite decisions for $\underline{c}$ and $\hat{\underline{c}}$, which is correct here. We consider case (b) now. In this case, it is *not* correct to make opposite decisions for $\underline{c}$ and $\hat{\underline{c}}$: Neither $\underline{c}$ nor $\hat{\underline{c}}$ are broken here.

Again, we can read this off from the Voronoi diagram $V(\{\xi;\ \eta;\ X_N;\ X_N - (1,0)\})$:



This motivates the following algorithm for cutting:

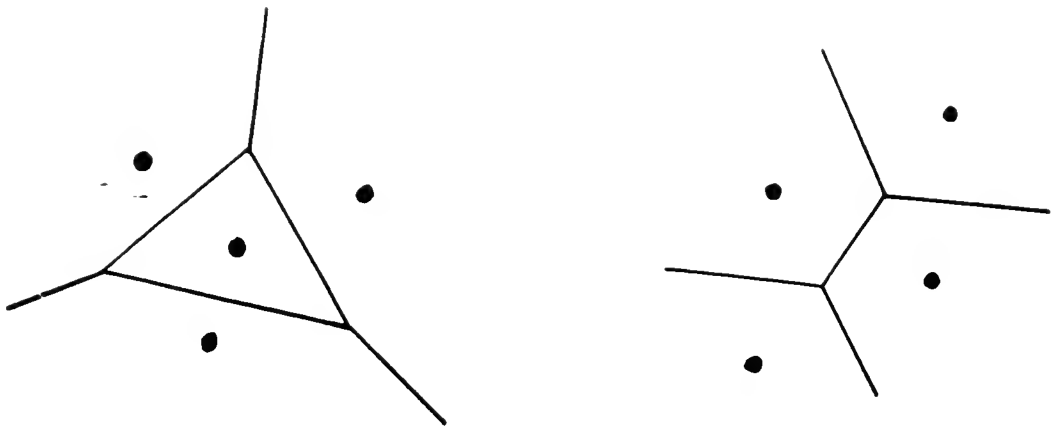
Algorithm 2.2:

For each corner c of $\hat{P}_N$, decide whether or not it lies in $[X_{N,1} - 0.5; X_{N,1} + 0.5] \times [X_{N,2} - 0.5; X_{N,2} + 0.5]$, using the following procedure. Decide whether c is closer to X_N than to $X_N + k_0$, $k_0 = \pm(1,0)$ or $k_0 = \pm(0,1)$ by computing

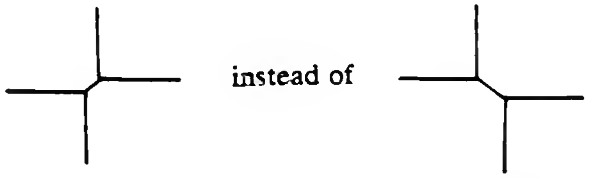
$$V(\{\xi; \eta; X_N; X_N + k_0\}), \tag{2.8}$$

where X_N, ξ, η are the generating points of c. If c is a corner in this diagram, it is at least as close to X_N as to $X_N + k_0$, otherwise it is broken by $X_N + k_0$. If $\tilde{c}$ is a corner of $\hat{P}_N$ generated by $X_N, \xi - k_0, \eta - k_0$, use the *same* diagram to decide whether or not $\tilde{c}$ is broken by $X_N - k_0$.

In our code, this procedure is simplified. The main simplification is a test whether cutting is necessary at all. This can safely be done by checking the coordinates of the corners of $\hat{P}_N$. In the vast majority of cases, no cutting is necessary. If cutting is required, we need to determine which of the four unordered triples of points in $\{\xi; \eta; X_N; X_N + k_0\}$ generate corners in the diagram (2.8). This is easy, since it is simple to decide for any given triple whether it forms a corner in (2.8). There are two or three corners in (2.8), as shown in the following figures.



Computing the topological structure of the Voronoi diagram is an ill-conditioned problem in the sense that the solution, the topological structure, changes discontinuously with the data, the given points. Therefore the difficulty which we have described and overcome should not be called numerical instability. We rejected the first version of the algorithm not because it computes the wrong topological structure, but because it computes two different structures at the same point on the torus. The improved version of the algorithm may still, due to rounding errors, compute the wrong topological structure:



This is unavoidable, but it fortunately does not matter in our application.

We remark that the *areas* of the Voronoi polygons depend on the given points in a continuous, in fact in a differentiable way; see section 3.

We finally describe two ways of finding a good starting point X for the search for the first broken corner. If the points in S are known to be roughly uniformly distributed in $[0;1]^2$, we can divide $[0;1]^2$ into n^2 square cells of equal size with $n^2 \approx N$ and create pointers from those cells to the points in S which they contain. A linked list is a good data structure for this purpose. In each cell, a pointer to one point in the cell is stored.

Then each point in the cell points to another in the cell, with the last point containing a stop code. We introduce the points cell by cell in the order indicated in the following figure:

n	n+1	..	N-n+1
..	n+2	..	N-n+2
2	..	..	..
1	2n	..	N

When introducing X_{k_0}, a good choice of X is then X_{k_0-1}. Using this method on random grids in $[0;1]^2$, we obtain the following CPU-times.

Table 2.1. CPU-times per point in msec on a VAX 11/780, 20 uniformly distributed random grids for each N.

N	worst case	best case	average
400	16	13	14
800	15	13	14
1200	15	14	14
1600	16	14	15
2000	16	15	15

The measured CPU-times support our assertion that the CPU-time per point is bounded independently of N.

In our fractional step method, we use this method at time 0. At later times, we use the following method. Assume that $V(\bar{S})$ has been constructed for a set $\bar{S}=\{\bar{X}_j + \underline{k} : j\in\{1,\cdots,N\}, \underline{k}\in Z^2\}$, where the distances $d(X_j,\bar{X}_j)$ are small, and $\bar{X}_j \in [0;1)^2$. Here d is defined as in (2.4). Let $j(k_0)$ be the index of a point in $\bar{S}$ whose polygon had a broken corner when $\bar{X}_{k_0}$ was inserted during the construction of $V(\bar{S})$. A suitable transla-

tion of the point $X_{j(k_0)}$ is then close to X_{k_0}. The indices $j(k_0) \in \{1;\cdots;k_0\}$ are therefore saved while constructing $V(\bar{S})$. In our application, S is the set of fluid markers and their periodic images at a given time step, and $\bar{S}$ is the corresponding set at the preceding time step.

This method is usually slightly more efficient than the first one. It also has the advantage of not requiring any renumbering of the points.

We conclude this section with some remarks about other possible ways of constructing periodic Voronoi diagrams.

It is, of course, possible to reduce the problem to nonperiodic Voronoi diagrams in the plane. Consider

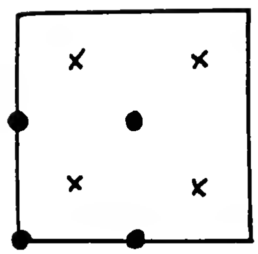
$$\hat{S} := \{X_j + k\colon 1 \le j \le N,\ k \in Z^2,\ |k_1| \le 1 \text{ and } |k_2| \le 1\}. \tag{2.9}$$

It is easy to see that

$$P(X_j,\hat{S}) = P(X_j,S) \text{ for all } j. \tag{2.10}$$

Since $\hat{S}$ contains 9N points, this procedure is quite inefficient. The definition of $\hat{S}$ can be modified to reduce the number of auxiliary points, but the algorithm then becomes complicated. It becomes impossible to search for the first broken corner using the second strategy, since the set of auxiliary points becomes dependent on the given configuration and therefore time dependent.

It may also be possible to work with the torus (2.3) and the metric (2.4) directly. The point insertion algorithm, however, cannot be used in literally the same way as in the plane. To see this, consider the step in which the coordinates of a new corner are computed after its three generating points X,Y,Z have been found. In the plane, this is done by computing the midpoint of the circle through X,Y,Z. On the torus T with metric d, there is no such unique midpoint, as is shown by the following example.



Each point "•" has the same distance from each point "x".

A periodic version of the algorithm used by Fritts (1981) appears to be an excellent alternative. This algorithm constructs the Delaunay triangulation rather than the Voronoi diagram and seems more efficient than our method.

However, we believe that an approach like the one described here may be more generally applicable because it makes use of the metric only and does not require an inner product or angles. For a version on the surface of a sphere see Augenbaum and Peskin (1985). A generalization to three-dimensional periodic Voronoi diagrams seems straightforward.

2.2 Some further algorithmic details and implementation of the method

We describe a Fortran 77 subroutine DIAGRAM which implements the method presented in section 2.1. The development of this subroutine began with a program written by C. Peskin for the construction of non-periodic Voronoi diagrams. Each of the following sections corresponds to one of the larger subroutines called by DIAGRAM.

2.2.1 Reordering fluid markers at time zero

We use integer indices

$$\mathrm{ICZ}(k,i),\quad i=1,\ldots,N,\quad k=0,1,2.$$

At t(=time)=0, these indices have the following meaning. When inserting the i-th point, the search for a broken corner begins with a corner of the polygon of the point

$$x_{\mathrm{ICZ}(0,i)} + \begin{pmatrix}\mathrm{ICZ}(1,i)\\ \mathrm{ICZ}(2,i)\end{pmatrix}.$$

See sections 2.2.3, 2.2.5 for the meaning and use of ICZ at t>0. At t=0, the following algorithm is used within DIAGRAM to generate ICZ.

Algorithm 2.3:

Step 1: Choose $n \approx \sqrt{N}$. Create pointers IBTOP$(k,l) \in \{1,\dots,N\}$, $(k,l) \in \{1,\dots,n\}^2$, and IPTOP$(i) \in \{1,\dots,N\}$, $i \in \{1,\dots,N\}$, with the following meaning. If IBTOP$(k,l)=0$, then there is no fluid marker in the cell $[(k-1)h; kh] \times [(l-1)h; lh]$, otherwise IBTOP$(k,l)$ is the index of one such marker. In that case, the list of all indices of markers in the cell is $i_1,\dots,i_r$, where $i_1=$IBTOP(k,l), $i_{j+1}=$IPTOP(i_j), and $j=r$ is the first integer for which IPTOP(i_j) is zero. The construction of these pointers proceeds marker by marker, so that each marker is associated with one and only one cell, even if it lies on the border between two or four cells. To reduce the required amount of testing, we use an array IN of size $n \times n$. IN(k,l) is the number of markers found so far in cell (k,l).

Step 2: Sort the fluid markers, following the ordering (k,l) = (1,1),...,(1,n), (2,n),...,(2,1), (3,1),... of the cells, with arbitrary ordering within each cell.

2.2.2 Initialization of variables

When introducing the first fluid marker, two corners with the same coordinate pair are generated. Information about these corners is stored on IPT, ICR, XC, YC, RAD2, as described in section 2.1.

The entries IPTCR(0,1),IPTCR(1,1),IPTCR(2,1) of the array IPTCR are initialized. Generally, at a stage of the construction at which i_0 points, $i_0 \le N$, have been introduced, IPTCR(m,i) is defined for $m=0,1,2$, $1 \le i \le i_0$. The corner

$$(\text{IPTCR}(1,i),\text{IPTCR}(2,i),\text{IPTCR}(0,i))$$

belongs to the i-th polygon at that stage. IPTCR(1,1),IPTCR(2,1) may be $\neq 0$.

A logical array FLAGB is initialized by FLAGB($i1,i2,j$)=.FALSE. for all

$$(i1,i2,j) \in \{-1,0,1\}^2\times\{1\cdots N\}.$$

We shall describe the use of FLAGB in section 2.2.4.

In sections 2.2.3 - 2.2.9, we shall give a detailed description of the insertion of all translations of the k_0-th point ($2\le k_0\le N$) into the diagram associated with the translations of the first k_0-1 points.

2.2.3 Determination of the corner at which the search for the first broken corner starts

We consider the case t>0. As before, we use the notation $\tilde{X}_1,\cdots,\tilde{X}_N$ for the positions of the points during the previous construction. Let $\mu_1,\cdots,\mu_N\in Z^2$ be such that $\|(\tilde{X}_k+\mu_k)-X_k\|$ is small (k=1,...,N). Within our fractional step method, the pairs μ_k are determined while moving the fluid markers. If X_k moves beyond the right boundary of $[0;1]^2$, for example, it is shifted by $\begin{pmatrix}\nu_{k,1}\\0\end{pmatrix}\in Z^2$ such that the shifted point lies in $[0;1]^2$, and $\mu_{k,1}:=-1$. Usually, $\nu_{k,1}$ will also equal -1.

ICZ is not newly generated but taken as generated during the previous call of DIAGRAM. It has the following meaning. When the k_0-th point $\tilde{X}_{k_0}$ was introduced during the previous time step, a corner of the polygon of the point

$$(\mathrm{ICZ}(1,k_0),\mathrm{ICZ}(2,k_0),\mathrm{ICZ}(0,k_0))$$

was broken.

Let X_{k_0} be the point to be inserted next. The basic index of the corner at which the search for a broken corner begins is

$$\mathrm{IC1}=\mathrm{IPTCR}(0,\mathrm{ICZ}(0,k_0)). \qquad (2.11)$$

The shift indices are

$$\mathrm{KX1}=\mathrm{IPTCR}(1,\mathrm{ICZ}(0,k_0))+\mathrm{ICZ}(1,k_0)+\mu_{k_0,1}-\mu_{\mathrm{ICZ}(0,k_0),1} \qquad (2.12)$$

and

$$KY1=IPTCR(2,ICZ(0,k_0))+ICZ(2,k_0)+\mu_{k_0,2}-\mu_{ICZ(0,k_0),2}. \qquad (2.13)$$

From (2.12),(2.13), it is clear that the pairs μ_k need to be stored only temporarily. They can be used to modify ICZ(1,k),ICZ(2,k) immediately and are not needed again.

Even though shift indices of corners broken by $X_{k_0} \in [0;1]^2$ always lie in $\{-1,0,1\}$, KX1 and KY1 may not belong to $\{-1,0,1\}$. The corner (KX1,KY1,IC1) is then unlikely to be a good starting point for the search for a broken corner. Lacking any better possibility, we redefine IC1=1,KX1=0,KY1=0 if (KX1,KY1) $\notin \{-1,0,1\}^2$, thus ensuring that the search for a broken corner can always be limited to corners with shift indices between -1 and 1. Such a limitation is useful because the number of storage locations needed for the array FLAGB (see section 2.2.4) is then 18N, which is the number of corners in nine copies of the unit square.

2.2.4 Search for a first broken corner

Beginning at (KX1,KY1,IC1), a breadth first search through the graph of corners is conducted to find a first broken corner. The array ICR contains the adjacency structure of this graph. We state the algorithm.

Algorithm 2.4:

Test whether the corner (KX1,KY1,IC1) is broken. If so, the algorithm terminates. Otherwise: Set

K1=1, LISTB(0,1)=IC1, LISTB(1,1)=KX1, LISTB(2,1)=KY1.

LISTB will be the list of tested corners. Set FLAGB(KX1,KY1,IC1)=.TRUE.. Generally, FLAGB(i1,i2,j) will be .TRUE. if and only if corner (i1,i2,j) has been tested.

Repeat the following steps:

IC=LISTB(0,K1), KX=LISTB(1,K1), KY=LISTB(2,K1). For each of the neighbors (KX2,KY2,IC2) of the corner (KX,KY,IC), test whether KX2$\in\{-1,0,1\}$ and KY2$\in\{-1,0,1\}$ and FLAGB(KX2,KY2,IC2)=.FALSE.. If all these conditions are satisfied, test whether the corner (KX2,KY2,IC2) is broken. If so, the algorithm terminates, otherwise (KX2,KY2,IC2) is appended to LISTB, and FLAGB(KX2,KY2,IC2) is set to .TRUE.. After all neighbors of the corner (KX,KY,IC) have been tested in this way, K1 is increased by 1.

Eventually, this algorithm will clearly find a broken corner.

2.2.5 Search for all broken corners

Once a first broken corner (KX1,KY1,IC1) has been found, the indices of the first generating point of this corner are stored as ICZ$(0,k_0)$, ICZ$(1,k_0)$, ICZ$(2,k_0)$. All entries in FLAGB are set back to .FALSE., using the list LISTB. Then an analogous search for all broken corners is performed, using LISTB to store broken corners and setting FLAGB(KX,KY,IC)=.TRUE. if (KX,KY,IC) is broken.

2.2.6 Computation of corners of $\hat{P}_{k_0}$

The corners of $\hat{P}_{k_0}$ are computed as described in section 2.1. The following information is generated.

NC0 = number of corners of $\hat{P}_{k_0}$.

XC0,YC0 = vectors of length NC0 containing the coordinates of those corners. Here and in all vectors of length NC0 described below, the ordering of the corners follows the boundary of the new polygon in counterclockwise direction. The use of this ordering could easily be avoided. This is important because there is no obvious analogous ordering in three dimensions.

RAD20 = vector of length NC0 containing the squares of their radii.

ITO = integer array. For $j=1,\cdots$,NC0, there is an edge from a corner broken by X_{k_0} to the unbroken corner (ITO(1,j),ITO(2,j),ITO(0,j)).

FLAGB0 = boolean vector of length 2N; FLAGB0(j)=.TRUE. if and only if the j-th stored corner is broken by some translation of X_{k_0}, i.e. if and only if there is a pair (i1,i2) such that FLAGB(i1,i2,j)=.TRUE. ($j \in \{1,\cdots,2k_0-2\}$).

LISTB0 = integer vector of length NBROKEN, giving the basic indices of all broken stored corners.

LISTP20,LISTP30 = integer arrays. For $j \in \{1,\cdots,\mathrm{NC0}\}$, the generating points of the j-th corner of the newly constructed polygon $\hat{P}_{k_0}$ are, in counterclockwise order, X_{k_0} and the points

$$
\begin{gathered}
(\mathrm{LISTP20}(1,j),\mathrm{LISTP20}(2,j),\mathrm{LISTP20}(0,j)), \\
(\mathrm{LISTP30}(1,j),\mathrm{LISTP30}(2,j),\mathrm{LISTP30}(0,j)).
\end{gathered}
$$

LISTL0 = vector of length NC0. For $j \in \{1,\cdots,\mathrm{NC0}\}$, the LISTL0(j)-th neighbor of corner (ITO(1,j),ITO(2,j),ITO(0,j)) is broken by X_{k_0}, and the j-th corner of $\hat{P}_{k_0}$ lies on the corresponding edge.

The arrays ICR, IPT, XC, YC, RAD2 which describe the periodic diagram are not yet changed.

2.2.7 Computation of corners of P_{k_0}

First we test whether all (XC0(j),YC0(j)), j=1,...,NC0, lie in the unit square centered at X_{k_0}. If so, no corners of $\hat{P}_{k_0}$ need to be cut off due to interaction with periodic images; see section 2.1. Otherwise we determine the corners to be cut off, using the following algorithm.

Algorithm 2.2b:

Initialize the entries of an integer vector IBR of length NC0 by IBR(j)=0 for all j. IBR(j)=1 (2,3,4) will mean that the j-th new corner (XC0(j),YC0(j)) is broken by the northern (western, southern, eastern) translation of X_{k_0}. It is easy to see that no other translation of X_{k_0} can break a corner of $\hat{P}_{k_0}$. Initialize the entries of a boolean vector DONE of length NC0 with values .FALSE..

For $j \in \{1,\cdots,N\}$, test whether there is a $j_0 \in \{1,\cdots,N\}$ such that the set consisting of the three generating points of the corner (XC0(j),YC0(j)) and $X_{k_0} - \binom{1}{0}$ is equal, up to translation, to the set consisting of the three generating points of the corner (XC0(j_0),YC0(j_0)) and $X_{k_0} + \binom{1}{0}$. See section 2.1 for illustrating figures.

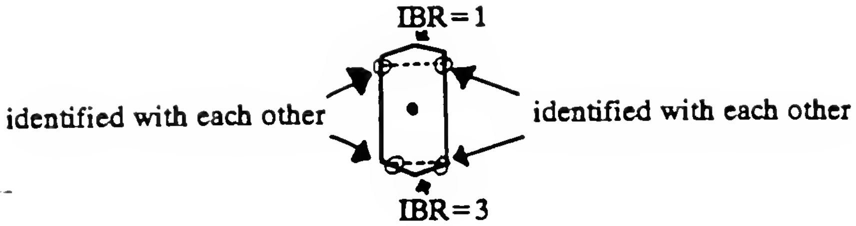
If such a j_0 does not exist, test whether XC0$<X_{k_0,1} - \frac{1}{2}$. If so, set IBR(j)=2, DONE(j)=.TRUE..

If such a j_0 does exist, compute the corresponding four-point Voronoi diagram (see section 2.1) and set DONE(j)=DONE(j_0)=.TRUE.. Set IBR(j)=2 if (XC0(j),YC0(j)) is found to be broken by $X_{k_0} - \binom{1}{0}$, IBR(j_0)=4 if (XC0(j_0),YC0(j_0)) is found to be broken by $X_{k_0} + \binom{1}{0}$.

Finally, test for all $j \in \{1,\cdots,N\}$ with DONE(j)=.FALSE. whether XC0(j)$>X_{k_0,1} + \frac{1}{2}$; if so, set IBR(j)=4.

Set all entries in DONE back to .FALSE. and search for corners (XC0(j),YC0(j)) broken by the northern and southern translations of X_{k_0} in exactly the same way.

If there are corners with IBR(j)=2, there are corners with IBR(j)=4. The same is true for corners with IBR(j)=1,3; see the following figure.



Because of rounding errors, this rule might be violated. Since the program relies on it later, it sets IBR(j) from i=1 (2,3,4) back to 0 if there are no indices j with IBR(j)=3 (4,1,2). This does not cause significant errors in the computed shapes of the polygons.

After having defined IBR, it is easy to create the list of corners of the polygon P_{k_0} of X_{k_0} in the new periodic diagram. We update

$$\text{NC0,XC0,YC0,LISTP20,LISTP30,RAD20,IBR,ITO,LISTL0.} \qquad (2.14)$$

NC0,XC0,YC0,LISTP20,LISTP30,RAD20 are updated in the obvious way. IBR(j)=0 now means that no translation of X_{k_0}, other than X_{k_0} itself, is among the generating points of the corner (XC0(j),YC0(j)). IBR(j)=1 (2,3,4) means that

$$X_{k_0}+\begin{pmatrix}0\\1\end{pmatrix} \quad \left(X_{k_0}+\begin{pmatrix}-1\\0\end{pmatrix},\ X_{k_0}+\begin{pmatrix}0\\-1\end{pmatrix},\ X_{k_0}+\begin{pmatrix}1\\0\end{pmatrix} \right)$$

is one of the generating points of (XC0(j),YC0(j)). ITO($\cdot$,j) does not have any meaning any more unless IBR(j)=0, nor does LISTL0 unless IBR(j)=0 and FLAGB0(ITO(0,j))=.FALSE..

Corners with IBR=1 (4) can be identified with corners with IBR=3 (2), as explained above. Each such pair represents only one corner in the new periodic diagram. Therefore NC0 is not yet the number of corners of X_{k_0} which will be stored. To prepare the insertion of the new corners into the new periodic diagram, indices IDENT(0,j), IDENT(1,j), IDENT(2,j), j=1,...,NC0, are computed. If IBR(j)$\notin$\{1,4\}, then IDENT(0,j)=j, IDENT(1,j)=0, IDENT(2,j)=0. If IBR(j)=1, then IDENT(1,j)=0, IDENT(2,j)=1, and IDENT(0,j) is defined as indicated in the previous figure. If

IBR(j)=4, then IDENT(1,j)=1, IDENT(2,j)=0, and IDENT(0,j) is defined analogously.

2.2.8 Computation of edges between corners of P_{k_0} and other corners

There are two cases to be distinguished here. If IBR(j)=0 and if FLAGB0(ITO(0,j)) is .FALSE., then the corner (ITO(1,j),ITO(2,j),ITO(0,j)) is the neighbor corner of the j-th corner of P_{k_0} in the new periodic diagram. If IBR(j)$\neq$0 or FLAGB0(ITO(0,j))=.TRUE., the neighbor corner of the j-th corner of P_{k_0} in the new diagram is a translation of one of the corners of P_{k_0}, say the l-th one. This translated corner is uniquely characterized by the fact that those of its generating points given by LISTP20,LISTP30 are the points which are also stored as LISTP20($\cdot$,j),LISTP30($\cdot$,j). It is found by searching through all corners of P_{k_0}. We perform this search only if IBR(j)=2 or IBR(j)=3, since corners with IBR=1 or IBR=4 will be identified with corners with IBR=2 or IBR=3.

2.2.9 Insertion of the new polygons into the periodic diagram

At this stage of the algorithm, LISTB0 contains a list of all basic indices of the broken corners. Theses indices serve as indices for the newly created corners. Exactly two additional basic indices are needed, namely $2k_0-1$ and $2k_0$. An integer vector ILISTB0 of length NC0 is created. It assigns to each index $j \in \{1,\ldots,\mathrm{NC0}\}$ in the circular chain of new corners a basic corner index in the new periodic diagram, i.e. one of the indices on LISTB0 or $2k_0-1,2k_0$. Since XC0(j),YC0(j) are not necessarily $\in[0;1]$, XC(ILISTB0(j)), YC(ILISTB0(j)) are not necessarily the same as XC0(j),YC0(j). The integer differences XC(ILISTB0(j)) − XC0(j), YC(ILISTB0(j)) − YC0(j) are stored as integer arrays ISHX,ISHY. Using ILISTB0, LISTP20, LISTP30, ICR0, ISHX, ISHY, RAD20, it is then easy to update IPT, ICR, IPTCR, and RAD2, completing the insertion of the k_0-th point and its translations into the periodic diagram.

3. Finite difference operators on irregular grids

We review the definitions of discrete Laplace, divergence and gradient operators due to Peskin (1979) and discuss their properties. Even though we use a terminology appropriate for 2 dimensions, the definitions and statements in this section are dimension independent.

We introduce the following notations. For $\underline{X} \in S_N$, let $V[\underline{X}]$ be the area of the Voronoi polygon $P(\underline{X},S_N) =: P[\underline{X}]$. For $\underline{Y} \in S_N$, $\underline{Y} \neq \underline{X}$, let $A[\underline{X},\underline{Y}]$ be the length of the common edge of $P(\underline{X},S_N)$ and $P(\underline{Y},S_N)$. $A[\underline{X},\underline{Y}] = 0$ if $P(\underline{X},S_N)$ and $P(\underline{Y},S_N)$ are not adjacent to each other. Our discrete operators couple two distinct points $\underline{X},\underline{Y}$ to each other only if they are neighbors, i.e. if $A[\underline{X},\underline{Y}] \neq 0$. In the following, we let $Nb[\underline{X}]$ be the set of neighbors of $\underline{X}$. By definition, $\underline{X} \notin Nb[\underline{X}]$.

3.1 Discrete Laplace operator

We define a discrete Laplace operator L by

$$V[\underline{X}](L\phi)(\underline{X}) := \sum_{\underline{Y} \neq \underline{X}} A[\underline{X},\underline{Y}] \frac{\phi(\underline{Y}) - \phi(\underline{X})}{\|\underline{Y}-\underline{X}\|} \quad \text{for } \underline{X} \in S_N . \tag{3.1}$$

This definition is motivated by the formula

$$\int_P \Delta\phi(\underline{x})d\underline{x} = \int_{\partial P} \frac{\partial\phi}{\partial \underline{n}}(\underline{x})ds \tag{3.2}$$

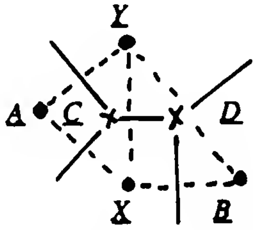
for smooth functions ϕ. The sum in (3.1) is formally infinite but contains only finitely many non-zero terms. Similar discretizations of the Laplace operator have been used in numerical computations for a long time; see, for example, MacNeal (1953). The following proposition was pointed out to us by B. Mercier (unpublished communication).

Proposition 3.1: The operator in (3.1), seen as a matrix, is the stiffness matrix obtained with piecewise linear finite elements on the Delaunay triangulation.

Proof:

Both matrices have zero row sums. Therefore we need to consider the non-diagonal entries only. The matrices have the same non-zero structure and are both symmetric. It therefore makes sense to talk about the coupling between $\underline{X}$ and $\underline{Y}$ ($\underline{X},\underline{Y} \in S_N,\ \underline{X} \neq \underline{Y}$). We compare the couplings in the matrix in (3.1) with those in the finite element stiffness matrix.

Without loss of generality, we assume that $\underline{X} = (0,0)$ and $\underline{Y} = (0,1)$. Let $\underline{A},\underline{B}$ be the points in S_N such that $\underline{A},\underline{X},\underline{Y}$ and $\underline{B},\underline{X},\underline{Y}$ are triangles in the Delaunay triangulation:



(solid: Voronoi diagram; dotted: Delaunay triangles)

Let $\underline{C}$ be the center of the circle through $\underline{A},\underline{X},\underline{Y}$, and let $\underline{D}$ be the center of the circle through $\underline{B},\underline{X},\underline{Y}$. $\underline{C}$ and $\underline{D}$ are corners in $V(S_N)$. The claim is then that the coupling between $\underline{X}$ and $\underline{Y}$ obtained with piecewise linear finite elements is $\| \underline{C} - \underline{D} \|$.

To show this, we first compute this coupling in terms of the coordinates of $\underline{A}$ and $\underline{B}$. The relevant finite element basis functions on the left triangle are

$$\phi_1(\underline{x}) \;=\; \frac{A_2-1}{A_1}x_1 - x_2 + 1 \;, \tag{3.3}$$

the basis function associated with $\underline{X}$, and

$$\phi_2(\underline{x}) \;=\; -\frac{A_2}{A_1}x_1 + x_2 \;, \tag{3.4}$$

the basis function associated with $\underline{Y}$. We multiply the gradients of these two functions, integrate the product over the left triangle and multiply by -1. The result is

$$-\frac{A_2^2 - A_2}{2A_1} - \frac{A_1}{2}. \tag{3.5}$$

The contribution from the right triangle is then, by symmetry,

$$\frac{B_2^2 - B_2}{2B_1} + \frac{B_1}{2}. \tag{3.6}$$

The difference in sign between (3.5) and (3.6) is due to the fact that the left triangle has the area $-A_1/2$, while the right triangle has the area $+B_1/2$. We have then shown that the finite element coupling between $\underline{X}$ and $\underline{Y}$ is

$$-\frac{A_2^2 - A_2}{2A_1} - \frac{A_1}{2} + \frac{B_2^2 - B_2}{2B_1} + \frac{B_1}{2}\,. \tag{3.7}$$

We next compute $||\underline{C} - \underline{D}||$.

$$\underline{C} = \left(\frac{1}{2}A_1 + \frac{A_2(A_2-1)}{2A_1}, 0.5\right) \tag{3.8}$$

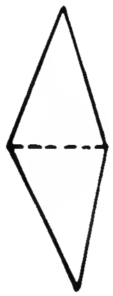
$$\underline{D} = \left(\frac{1}{2}B_1 + \frac{B_2(B_2-1)}{2B_1}, 0.5\right) \tag{3.9}$$

Therefore the square of (3.7) equals the square of $||\underline{C} - \underline{D}||$.

To conclude the proof of our assertion, it remains to be shown that (3.7) is positive. We notice that the foregoing computations do not make any use of the fact that we are using Delaunay triangulations and Voronoi diagrams. However, all off-diagonal elements of the finite element Laplacian obtained with a triangulation τ are positive if and only if τ is a Delaunay triangulation.

We outline the proof of this well-known result.

A triangulation τ is called locally equiangular if it has the following property. If T_1 and T_2 are adjacent triangles in τ whose union Q is a convex quadrilateral, the sum of the angles in Q which are cut by the common edge of T_1 and T_2 is larger than 180^o.



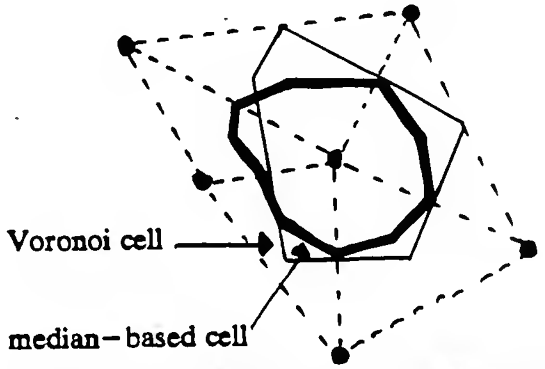
but not

It is obvious that Delaunay triangulations are locally equiangular. Sibson (1978) showed that the converse is also true.

A simple calculation shows that all off-diagonal elements of the finite element Laplacian obtained with a triangulation τ are positive if and only if τ is locally equiangular; see Fritts (1981). We note that this condition is also equivalent to the diagonal dominance of the finite element stiffness matrix, since this matrix has the zero row sum property.

This concludes the proof of proposition 3.1. ■

In spite of proposition 3.1, a discretization of the Poisson equation based on (3.1) is not a common finite element discretization. The difference lies in the right-hand sides. Let f be a given continuous right-hand side. In the finite element method, the discrete right-hand side consists of integrals of the form $\int_{\Omega} \tilde{f}(\underline{x})\beta(\underline{x})dx$, where $\tilde{f}$ is an approximation of f and β is a basis function of the finite element space. When using piecewise linear elements, the common choice of $\tilde{f}$ is the piecewise linear interpolant of f. The value in $\underline{X}$ of the discrete right-hand side is then a weighted sum of values of f in neighbors of $\underline{X}$. The sum $M[\underline{X}]$ of the weights (elements of the mass matrix) is one third times the sum of the areas of the triangles to which $\underline{X}$ belongs. Using (3.1), one is lead to using $V[\underline{X}]f(\underline{X})$ as the value of the discrete right-hand side in $\underline{X}$. $V[\underline{X}]$ and $M[\underline{X}]$ differ from each other in general, but they are, of course, equal on the average. $M[\underline{X}]$ is the area of a cell suggested by Dukowicz (1981) and others, which is obtained by joining the centroids of the triangles with the midpoints of their sides. We draw the two different cells in one figure:



(The median−based cell need not be convex.)

We now prove that the operator L is weakly consistent to first order with the continuous Laplacian. This proof, formulated as a proof of weak consistency of the discrete divergence operator D defined in section 3.2, is due to C. Peskin and is contained in Börgers and Peskin (1985). We define

$$h := \max_{X} \operatorname{diam}(P[X]) \quad . \tag{3.10}$$

Proposition 3.2: If

$$N \le O(\frac{1}{h^2}), \tag{3.11}$$

then

$$\sum_k \psi(X_k) L\phi(X_k) V[X_k] = \int_{[0;1]^2} \psi(x)\Delta\phi(x)dx + O(h) \tag{3.12}$$

for arbitrary smooth, periodic scalar functions ϕ and ψ.

Proof:

We approximate the difference to be estimated by

$$\sum_k \psi(X_k)(L\phi)(X_k)V[X_k] \quad - \quad \sum_k \psi(X_k) \int_{P[X_k]} \Delta\phi(x)dx \tag{3.13}$$

Since ψ is uniformly continuous, the error which we have introduced is only $O(h)$. We next apply Green's formula (3.2) and its discrete counterpart (3.1) and obtain

$$\sum_k \sum_{Y \neq X_k} \psi(X_k) \left[A[X_k, Y] \frac{\phi(Y) - \phi(X_k)}{\|Y - X_k\|} \quad - \int_{P[X_k] \cap P[Y]} \frac{\partial \phi}{\partial n}(x) ds \right] . \tag{3.14}$$

Since the expression in the bracket is of order $O(h^2)$, and because of (3.11), we seem to obtain an upper bound of size $O(1)$ now. However, observe that the roles of X_k and Y in (3.14) may be reversed, therefore (3.14) is equal to

$$- \sum_k \sum_{Y \neq X_k} \psi(Y) \left[A[X_k, Y] \frac{\phi(Y) - \phi(X_k)}{\|Y - X_k\|} \quad - \int_{P[X_k] \cap P[Y]} \frac{\partial \phi}{\partial n}(x) ds \right] . \tag{3.15}$$

Here $\underline{n}$ always denotes the normal on $P[\underline{X}_k] \cap P[\underline{Y}]$ which is exterior with respect to $P[\underline{X}_k]$. Since (3.14) and (3.15) are equal, they are also both equal to the average of the two expressions,

$$\frac{1}{2}\sum_k \sum_{\underline{Y} \neq \underline{X}_k} (\psi(\underline{X}_k)-\psi(\underline{Y})) \left[A[\underline{X}_k,\underline{Y}] \frac{\phi(\underline{Y}) - \phi(\underline{X}_k)}{\|\underline{Y} - \underline{X}_k\|} \quad - \int_{P[\underline{X}_k] \cap P[\underline{Y}]} \frac{\partial \phi}{\partial \underline{n}}(\underline{x}) ds \right], \quad (3.16)$$

which is $O(h)$. ■

Assumption (3.11) is not always satisfied. As an example, consider a case in which all $\underline{X}_k$ lie on a vertical or horizontal straight line. Then $h=O(1)$, independently of N. (3.11) is a non-degeneracy assumption on the polygons.

We give a second, shorter proof of (3.12). Because of proposition 3.1, (3.12) is equivalent to

$$a(\phi_h,\psi_h) = a(\phi,\psi) + O(h), \quad (3.17)$$

where $a(\cdot,\cdot)$ is the bilinear form

$$a(\phi,\psi) := \int_{[0;1]^2} \underline{\nabla}\phi(\underline{x})\cdot\underline{\nabla}\psi(\underline{x}) d\underline{x} \quad (3.18)$$

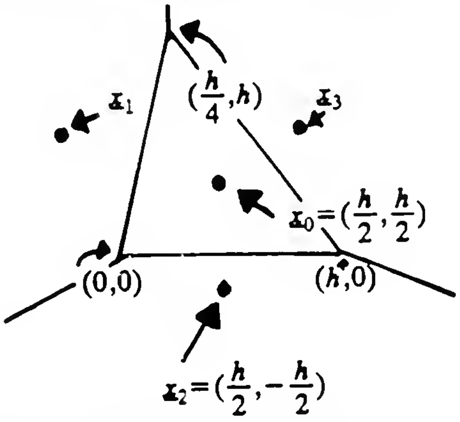
associated with the Laplace operator and ϕ_h,ψ_h are the piecewise linear interpolants of ϕ,ψ with respect to the Delaunay triangles. (3.17) is well known from the theory of interpolation in Sobolev spaces needed in finite element convergence theory; see Ciarlet (1978). The non-degeneracy assumption on the triangulation which is needed for this argument is, however, neither necessary nor sufficient for (3.11) to hold.

We gave the first argument for two reasons. First, it is independent of the connection with the finite element method and therefore applicable even if the control areas are not Voronoi polygons but areas defined in some different way. Second, it is applicable to discretizations of differential operators other than the Laplace operator; see below for the case of the divergence operator.

Proposition 3.3: L is not pointwise consistent with the Laplace operator.

Proof:

We give a counterexample. Consider the following Voronoi diagram.



We use Taylor's theorem to expand the expression

$V[x_0]L\phi(x_0) =$

$$\frac{\phi(x_1)-\phi(x_0)}{||x_1-x_0||}\sqrt{\frac{17}{16}}h + \frac{\phi(x_3)-\phi(x_0)}{||x_3-x_0||}\sqrt{\frac{25}{16}}h + \phi(x_2)-\phi(x_0)$$

around x_0. The coefficient before the mixed derivative $\frac{\partial^2\phi}{\partial x_1 \partial x_2}$ in this expansion is $-\frac{24}{425}h^2$, which proves the inconsistency of L. Nothing is special about this configuration. Almost any configuration without symmetry properties could serve as a counterexample. ■

L is pointwise consistent of order 0 in the following sense.

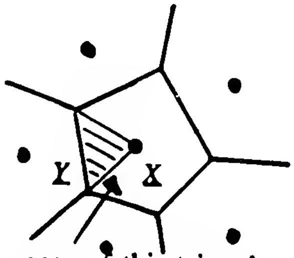
Proposition 3.4: For a fixed smooth periodic function ϕ, $L\phi$ is bounded independently of the fluid marker configuration.

Proof:

$$\begin{aligned}
|L\phi(X)| &= \left|\frac{1}{V[X]}\sum_{Y \neq X}\frac{\phi(Y)-\phi(X)}{||Y-X||}A[X,Y]\right| \\
&= \left|\frac{1}{V[X]}\sum_{Y \neq X}\frac{\phi(Y)-\phi(X)}{||Y-X||}A[X,Y] - \frac{1}{V[X]}\sum_{Y \neq X}\nabla\phi(X)\cdot\frac{Y-X}{||Y-X||}A[X,Y]\right| \\
&\leq \frac{C}{V[X]}\sum_{Y \neq X}||Y-X||A[X,Y]
\end{aligned}$$

for some constant C depending on ϕ only. Now notice that

$$\sum_{\underline{Y} \neq \underline{X}} A[\underline{X},\underline{Y}]||\underline{Y}-\underline{X}|| = 4V[\underline{X}].$$



area of this triangle:

$$\frac{1}{4}A[\underline{X},\underline{Y}]||\underline{X}-\underline{Y}||$$

This concludes the proof of proposition 3.4. ■

For later reference, we define the discrete L^2-product by

$$(\phi,\psi) := \sum_k \phi(\underline{X}_k)\psi(\underline{X}_k)V[\underline{X}_k] \tag{3.19}$$

for scalar functions and

$$(\underline{u},\underline{v}) := \sum_k \underline{u}(\underline{X}_k)\cdot\underline{v}(\underline{X}_k)V[\underline{X}_k]. \tag{3.20}$$

for vector fields.

Also for later use, we state and prove the following simple algebraic properties of L.

Proposition 3.5: L is symmetric and negative semidefinite in the inner product (3.19). The kernel of L is the set of constants.

Proof:

$$(L\phi,\psi) = \sum_k \sum_{\underline{Y} \neq \underline{X}_k} A[\underline{X}_k,\underline{Y}]\frac{\phi(\underline{Y}) - \phi(\underline{X}_k)}{||\underline{Y} - \underline{X}_k||}\psi(\underline{X}_k). \tag{3.21}$$

Reversing the roles of $\underline{X}_k$ and $\underline{Y}$, we obtain

$$(L\phi,\psi) = -\sum_k \sum_{\underline{Y} \neq \underline{X}_k} A[\underline{X}_k,\underline{Y}]\frac{\phi(\underline{Y}) - \phi(\underline{X}_k)}{||\underline{Y} - \underline{X}_k||}\psi(\underline{Y}). \tag{3.22}$$

Taking the average of (3.21) and (3.22), we obtain

$$(L\phi,\psi) = -\frac{1}{2}\sum_k \sum_{Y \neq X_k} A[X_k,Y]\frac{\phi(Y) - \phi(X_k)}{||Y - X_k||}(\psi(Y) - \psi(X_k)), \tag{3.23}$$

from which the assertions follow. ■

3.2 Discrete divergence operator

We turn to the definition of the discrete divergence operator,

$$V[X]Du(X) := \sum_Y u(Y)\cdot\frac{\partial V[X]}{\partial Y} \text{ for } X \in S_N . \tag{3.24}$$

Here $\frac{\partial V[X]}{\partial Y}$ is the gradient of $V[X]$ with respect to Y, keeping all other points fixed; see proposition 3.7. We also use the notation $D =: D_1$ to distinguish D from a different discretization D_2 of $\nabla\cdot$ introduced below.

If $(X_1,\cdots,X_N) = (X_1(t),\cdots,X_N(t))$ are points moving at velocity u, then

$$\frac{d}{dt}V[X_j(t)] = V[X_j(t)]Du(X_j(t)) \tag{3.25}$$

for all j, in particular:

Proposition 3.6: If $X_1(t),\cdots,X_N(t)$ are points in space which move in a discretely divergence-free velocity field, the Voronoi polygons maintain their areas exactly.

However, the Voronoi polygons do change their areas, usually even quite drastically, if the points are moved in a *continuously* divergence-free field. As pointed out by J. Dukowicz (personal communication), this is even true for large numbers of fluid markers and small diameters of the Voronoi polygons. We give an example which proves the correctness of this observation. Consider a flow of the form

$$u = Ax \tag{3.26}$$

with a constant matrix A. Let A have trace zero, so that (3.26) describes an incompressible flow. One could think of (3.26) as a local approximation to a real flow with a

stagnation point. Let a set of fluid markers move in this flow and consider the corresponding Voronoi diagrams. The Voronoi polygons do not necessarily maintain their areas. As a specific example, consider a flow where

$$A = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} : \tag{3.27}$$

If one of the markers is at the origin $\underline{x}=\underline{0}$ at time 0, it does not move. Its Voronoi polygon at later times depends on markers which are far away from it at time 0 and will therefore usually not maintain its area. If the coordinates of all fluid markers at time 0 are multiplied by ϵ, then the resulting Voronoi diagrams are just scaled by the same factor ϵ.

In summary, this example shows that Voronoi polygons of a set of points moving in a continuously divergence free flow may change their areas over a fixed time by a factor which does not converge to 1 when the set is made finer in the sense that h, as in (3.10), tends to zero. Furthermore, the example shows that the same statement is true for all control areas which are just scaled by ϵ when the given set of points is scaled by ϵ.

An explicit formula for $\frac{\partial V[\underline{X}]}{\partial \underline{Y}}$ was derived by Peskin (1979). For completeness, we present a (slightly different) derivation here.

Proposition 3.7: The mapping from $\{(\underline{X}_1,\cdots,\underline{X}_N)\in(0;1)^{2N} : \underline{X}_i \neq \underline{X}_j \text{ for } i\neq j\}$ into R^N which maps the cartesian coordinates of $\underline{X}_1,\cdots,\underline{X}_N$ onto the areas $V[\underline{X}_1],\cdots,V[\underline{X}_N]$ is continuously differentiable. The partial derivatives are given by

$$\frac{\partial V[\underline{X}]}{\partial \underline{Y}} = \frac{\underline{Y}-\underline{C}[\underline{X},\underline{Y}]}{||\underline{Y}-\underline{X}||} A[\underline{X},\underline{Y}] . \tag{3.28}$$

$||\frac{\partial V[\underline{X}]}{\partial \underline{Y}}||$ is not bounded for $\underline{Y} \to \underline{X}$.

Proof:

We distinguish the following three cases for a given $\underline{Y}$:

i) $\underline{X} = \underline{Y}$

ii) $\underline{X} \in Nb[\underline{Y}]$ (so $\underline{X} \neq \underline{Y}$)

iii) $\underline{X} \neq \underline{Y}$ and $\underline{X} \notin Nb[\underline{Y}]$.

We first observe

$$\frac{\partial V[\underline{X}]}{\partial \underline{Y}} = 0 \text{ in case (iii)} . \tag{3.29}$$

Case (i) may be reduced to case (ii) using (3.29) and one of the two formulae

$$\sum_{\underline{X}} \frac{\partial V[\underline{X}]}{\partial \underline{Y}} = 0. \tag{3.30}$$

$$\sum_{\underline{Y}} \frac{\partial V[\underline{X}]}{\partial \underline{Y}} = 0. \tag{3.31}$$

(3.29) ensures that the sums (3.30), (3.31) are finite. (3.30) can then be derived in the following way. The summation can be restricted to the points in a finite number m of periods. $\sum_{\underline{X}} \frac{\partial V[\underline{X}]}{\partial \underline{Y}}$ is the rate of change of the total area of the Voronoi polygons associated with those points. This total area equals m. Therefore (3.30) follows.

To see (3.31), let the points $\underline{X}_1, \cdots, \underline{X}_N$ all move at the same constant velocity $\underline{u}$. Then

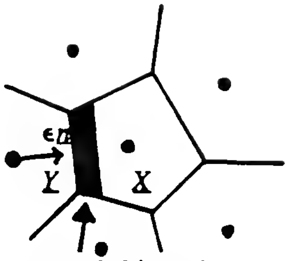
$$\sum_{\underline{Y}} \frac{\partial V[\underline{X}]}{\partial \underline{Y}} \cdot \underline{u} \tag{3.32}$$

is the rate of change of $V[\underline{X}]$. But $V[\underline{X}]$ clearly does not change at all, $P[\underline{X}]$ is just translated. Since $\underline{u}$ is arbitrary, (3.31) follows. Both (3.30) and (3.31) can be used to represent the diagonal derivatives in terms of the off-diagonal ones. The two representations are different from each other.

We turn to case (ii) now. We first consider the derivative of $V[\underline{X}]$ in the direction $\frac{\underline{Y}-\underline{X}}{\|\underline{Y}-\underline{X}\|}$. This derivative is

$$\frac{1}{2}A[\underline{X},\underline{Y}] \tag{3.33}$$

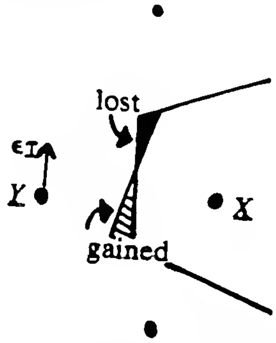
as can be seen from the following figure:



area of this strip:

$$\frac{1}{2}\epsilon A[\underline{X},\underline{Y}]+O(\epsilon^2)$$

We next consider the derivative in the direction parallel to the face $P[\underline{X}]\cap P[\underline{Y}]$.



Up to corrections of area $O(\epsilon^2)$, both the gained triangle and the lost triangle are similar to the triangle formed by $\underline{X},\underline{Y},\underline{Y}+\epsilon\underline{t}$. The scaling factors are

$$\frac{-||\frac{1}{2}(\underline{X}+\underline{Y}) - \underline{C}[\underline{X},\underline{Y}]|| + \frac{1}{2}A[\underline{X},\underline{Y}]}{||\underline{X}-\underline{Y}||} \tag{3.34}$$

for the gained triangle and

$$\frac{||\frac{1}{2}(\underline{X}+\underline{Y}) - \underline{C}[\underline{X},\underline{Y}]|| + \frac{1}{2}A[\underline{X},\underline{Y}]}{||\underline{X}-\underline{Y}||} \tag{3.35}$$

for the lost triangle. Therefore the total area change is

$$\frac{\epsilon}{2}||\underline{X}-\underline{Y}|| \left[\frac{(\frac{1}{2}A[\underline{X},\underline{Y}] - ||\frac{1}{2}(\underline{X}+\underline{Y}) - \underline{C}[\underline{X},\underline{Y}]||)^2}{||\underline{X}-\underline{Y}||^2} - \frac{(\frac{1}{2}A[\underline{X},\underline{Y}] + ||\frac{1}{2}(\underline{X}+\underline{Y}) - \underline{C}[\underline{X},\underline{Y}]||)^2}{||\underline{X}-\underline{Y}||^2} \right], \tag{3.36}$$

which is

$$-\frac{\epsilon A[X,Y]\|\frac{1}{2}(X+Y)-C[X,Y]\|}{\|X-Y\|} \quad . \tag{3.37}$$

Area is lost moving in the direction of the center $C[X,Y]$, and area is gained moving away from $C[X,Y]$. From what we have shown, it follows that

$$\frac{\partial V[X]}{\partial Y} = \frac{1}{2}A[X,Y]\cdot\frac{Y-X}{\|Y-X\|} + \frac{A[X,Y]}{\|X-Y\|}\left(\frac{1}{2}(X+Y)-C[X,Y]\right), \tag{3.38}$$

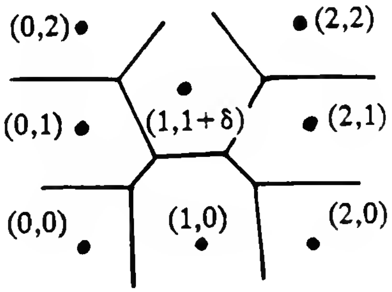
which is equivalent with (3.28). ∎

We gave a proof for proposition 3.7 in two space dimensions. However, eq. (3.28) is correct in all dimensions, if $C[X,Y]$ is defined as the center of mass of the face between X and Y. Peskin's derivation of (3.28) shows this; see Börgers and Peskin (1985).

We notice that the mapping

$$(X_1,\cdots,X_N) \rightarrow (V[X_1],\cdots,V[X_N])$$

is not everywhere twice continuously differentiable. To see this, consider the following figure.



With $X:=(1,0)$ and $Y:=(1,1+\delta)$, we have $A[X,Y]=1-\delta$ for $\delta\in[0;1]$ here, and $A[X,Y]=0$ for $\delta\geq 1$. Therefore the normal component of $\frac{\partial V[X]}{\partial Y}$ is not everywhere differentiable (see (3.38)).

To prove the weak consistency of D with the continuous divergence operator, we

first observe that the argument which was used to prove (3.12) can be extended in the following way.

Let Λ be a linear differential operator. Assume that a formula of the form

$$\int_P \Lambda\phi(\underline{x})d\underline{x} = \int_{\partial P} \sigma\phi(\underline{x})ds \tag{3.39}$$

holds for all smooth functions ϕ, where σ is a linear differential operator on ∂P, possibly of order zero. In the case where Λ is the Laplace operator, $\sigma\phi = \frac{\partial\phi}{\partial \underline{n}}$. Since σ depends on P, we also use the notation

$$\sigma = \sigma_P .$$

We discretize Λ by

$$V[\underline{X}_k]L\phi(\underline{X}_k) := \sum_{\underline{Y} \neq \underline{X}_k} I[\underline{X}_k,\underline{Y}], \tag{3.40}$$

where

$$I[\underline{X}_k,\underline{Y}] = \int_{P[\underline{X}_k]\cap P[\underline{Y}]} \sigma\phi(\underline{x})ds + O(h^2). \tag{3.41}$$

If we replaced $O(h^2)$ by $O(h^3)$ here, pointwise consistency of first order of L and Λ would follow immediately. We assume in addition that

$$\int_{P[\underline{X}_k]\cap P[\underline{Y}]} \sigma_{P[\underline{X}_k]}\phi(\underline{x})ds = - \int_{P[\underline{X}_k]\cap P[\underline{Y}]} \sigma_{P[\underline{Y}]}\phi(\underline{x})ds \tag{3.42}$$

$$I[\underline{X},\underline{Y}] = - I[\underline{Y},\underline{X}] + O(h^3) \tag{3.43}$$

Proposition 3.8: (3.39)-(3.43) together with the non-degeneracy assumption (3.11) imply the weak consistency of L with Λ.

The proof exactly duplicates that of proposition 3.2. Analogous propositions hold if Λ and L act on vectors rather than scalars, or if the $P[\underline{X}]$ are not the Voronoi polygons but some different control areas with mutually disjoint interiors, covering the area of interest.

From proposition 3.8, we now conclude the weak consistency of D.

Proposition 3.9: Let the non-degeneracy assumption (3.11) be satisfied. Then the operator D is weakly consistent with the continuous divergence operator, i.e.

$$\sum_{k} \phi(\underline{X}_k) D\underline{u}(\underline{X}_k) V[\underline{X}_k] = \int_{[0;1]^2} \phi(\underline{x}) \underline{\nabla}\cdot\underline{u}(\underline{x}) d\underline{x} + O(h) \tag{3.44}$$

for any smooth periodic (scalar and vector) functions ϕ and $\underline{u}$.

Proof:

$$\int_P \underline{\nabla}\cdot\underline{u} d\underline{x} = \int_{\partial P} \underline{u}(\underline{x})\cdot\underline{n} ds, \tag{3.39b}$$

i.e.

$$\sigma\underline{u} = \underline{u}\cdot\underline{n} \ ,$$

$I[\underline{X},\underline{Y}] =$

$$\left[\frac{1}{2}(u(\underline{X})+u(\underline{Y}))A[\underline{X},\underline{Y}]\frac{\underline{Y}-\underline{X}}{||\underline{Y}-\underline{X}||}\right] + \left[\frac{\underline{u}(\underline{Y})-\underline{u}(\underline{X})}{||\underline{Y}-\underline{X}||}A[\underline{X},\underline{Y}](\frac{1}{2}(\underline{X}+\underline{Y})-\underline{C}[\underline{X},\underline{Y}])\right] . \tag{3.40b}$$

The first summand in (3.40b) originates from the normal component of (3.38), the second from the tangential component. In the first summand, one obtains $-u(\underline{X})$ rather than $u(\underline{X})$ from (3.38) and (3.31). To change the sign is correct because the integral over $\partial P[\underline{X}]$ of the exterior unit normal is zero. We obtain

$$I[\underline{X}_k,\underline{Y}] = \int_{P[\underline{X}_k]\cap P[\underline{Y}]} \underline{u}(\underline{x})\cdot\underline{n} ds + O(h^2). \tag{3.41b}$$

This relation would remain true if we dropped the second summand in (3.40b), which is itself of order $O(h^2)$. We thus obtain an alternative discretization D_2 of $\underline{\nabla}\cdot$. D_2 can also be derived by discretizing the divergence theorem on the Voronoi polygons in the most obvious way. See section 5 for numerical results concerning the comparison between $D=D_1$ and D_2.

Furthermore we obtain

$$\int_{P[\underline{X}_k]\cap P[\underline{Y}]} \sigma_{P[\underline{X}_k]}\underline{u}(\underline{x}) ds = - \int_{P[\underline{X}_k]\cap P[\underline{Y}]} \sigma_{P[\underline{Y}]}\underline{u}(\underline{x}) ds, \tag{3.42b}$$

$$I[\underline{X},\underline{Y}] = -I[\underline{Y},\underline{X}] \text{ (exactly)}. \tag{3.43b}$$

So (3.44) follows. ■

Proposition 3.10: D and D_2 are pointwise inconsistent.

Proof:

The inconsistency of D was first shown by M. McCracken (unpublished). We use the same counterexample as for L. We consider

$$V[\underline{x}_0]D\underline{u}(\underline{x}_0) = \frac{\underline{u}(\underline{x}_1)-\underline{u}(\underline{x}_0)}{\|\underline{x}_1-\underline{x}_0\|}\left(\underline{x}_1-\begin{pmatrix}0.125\\0.5\end{pmatrix}h\right)\sqrt{\frac{17}{16}}h +$$

$$\frac{\underline{u}(\underline{x}_3)-\underline{u}(\underline{x}_0)}{\|\underline{x}_3-\underline{x}_0\|}\left(\underline{x}_3-\begin{pmatrix}0.625\\0.5\end{pmatrix}h\right)\sqrt{\frac{25}{16}}h + \left(\underline{u}(\underline{x}_2)-\underline{u}(\underline{x}_0)\right)\begin{pmatrix}0\\-0.5\end{pmatrix}h. \tag{3.45}$$

The coefficient before $\dfrac{\partial u_1}{\partial x_2}$ in the Taylor expansion of (3.45) is $-\dfrac{24}{425}h^2$.

The pointwise inconsistency of D_2 can be shown with the same counterexample. The coefficient before $\dfrac{\partial u_1}{\partial x_2}$ becomes $-\dfrac{12}{425}h^2$. ■

We give a second proof of the inconsistency of D. This argument is a reformulation of our previous considerations concerning the example (3.26), (3.27) and shows a statement slightly more general than proposition 3.10. As mentioned above, the area of the Voronoi polygon of a marker at (0,0) will, in general, not be constant. This implies that there are fluid marker configurations for which $D\underline{u}(0,0)\neq 0$, if $\underline{u}(\underline{x})=\begin{pmatrix}x_2\\0\end{pmatrix}$. Take such a configuration and scale it by ϵ. This causes scaling of the gradients $\dfrac{\partial V[(0,0)]}{\partial \underline{Y}}$ by ϵ and scaling of $V[(0,0)]$ by ϵ^2. Because of $\underline{u}(\epsilon\underline{Y})=\epsilon\underline{u}(\underline{Y})$, $Du(0,0)$ remains unchanged. The same argument shows the pointwise inconsistency of any discrete divergence defined as the relative rate of change of the area of a cell which is scaled by ϵ if the marker configuration is scaled by ϵ.

3.3 Discrete gradient operator

We define $\underset{\sim}{G}$ as the negative adjoint of D with respect to the discrete L^2 inner product (3.19). We also define $\underset{\sim}{G}_2$ as the negative adjoint of D_2 with respect to (3.19).

Proposition 3.11: $\underset{\sim}{G}$ and $\underset{\sim}{G}_2$ are weakly consistent with the continuous gradient operator.

Proof:

This follows from the weak consistency of D, D_2 by integrating by parts. ■

From (3.30), it follows that the kernel of $\underset{\sim}{G}$ contains the constants. (Similarly, it follows from (3.31) that the kernel of D contains the constant vector fields.) Therefore $(D\underset{\sim}{u},1)=0$ for all $\underset{\sim}{u}$, and $L\phi=D\underset{\sim}{u}$ always has a solution ϕ.

The kernel of $\underset{\sim}{G}$ may contain non-constant functions. If, for example, $\underset{\sim}{X}_1,\cdots,\underset{\sim}{X}_N$ lie on a square grid, and if N is the square of an even integer, then the kernel of $\underset{\sim}{G}$ is four-dimensional, for $\underset{\sim}{G}$ is then the discretization of $\underset{\sim}{\nabla}$ with central difference quotients. We have been unable to find a general estimate of the dimension of the kernel of $\underset{\sim}{G}$.

4. Solution of discrete Helmholtz equations on irregular grids

In this section, we consider the fast numerical solution of

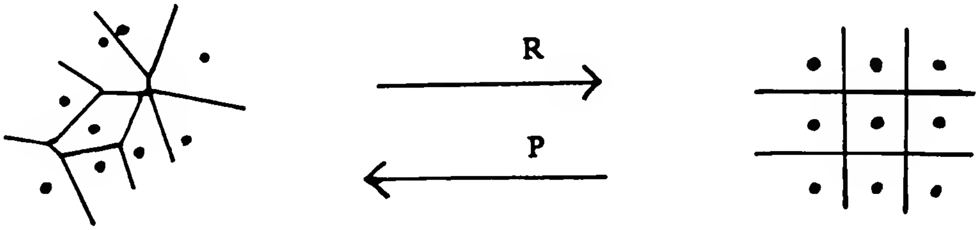
$$-L\phi + c\phi = f \quad (c \geq 0) \ . \tag{4.1}$$

We use a two-level iteration which is identical with the well-known multigrid correction cycle, replacing the coarsening of the grid by regularization.

Consider a regular grid of the form

$$\Gamma^h = \{((i + 0.5)h, (j + 0.5)h) : i,j \in Z\} \tag{4.2}$$

with $h = 1/n$ and $n \approx \sqrt{N}$. Let R be a linear interpolation operator which maps functions defined on S_N onto functions defined on Γ^h. Let P be a linear interpolation operator in the opposite direction.



We introduce the abbreviations

$$A := -L + cI \text{ and} \tag{4.3}$$

$$B := \text{discretization of } -\Delta + \text{cI on } \Gamma^h\text{, using the standard } 5\text{-point operator.} \tag{4.4}$$

We consider the following simple iterative method for (4.1):

$$\phi^{n+1} := \phi^n + PB^{-1}R(f - A\phi^n) \ . \tag{4.5}$$

If c=0, B^{-1} does not exist. $B^{-1}r$ is then obtained as follows. Subtract the constant component from r, find some solution v of $Bv = r$, and subtract the constant component from v. That is, B^{-1} is the pseudoinverse of B.

If P and R are chosen in a simple, straightforward way, the iteration (4.5) converges very poorly if at all. The reason is that P and R introduce smoothing, which

prevents highly oscillatory errors from converging fast.

Therefore (4.5) has to be supplemented by a method which is efficient on such errors. We choose a suitable relaxation method for this, obtaining the two-level cycle

Step 1: ν_1 *relaxation sweeps*
Step 2: (4.5) (4.6)
Step 3: ν_2 *relaxation sweeps.*

ν_1 and ν_2 are small integers, typically 1 or 2. To solve problems with B, as required in (4.5), we use the Fast Fourier Transform. This will be the only piece in our fractional step method which costs O(NlogN) operations rather than O(N) operations per time step. Instead of the Fast Fourier Transform, a very fast approximate solver, for example a multigrid solver, requiring only O(N) operations could be used. Such a replacement often does not have any significant influence on the convergence factors; see Stüben and Trottenberg (1982).

We have to choose P, R and the relaxation scheme. We discuss the choice of P and R first. We list some desirable properties:

CP: P preserves constants: $P(1)(X_j) = 1$ for all j.

CR: R preserves constants: $R(1)(x) = 1$ for all $x \in \Gamma^h$.

I0P: P preserves zero mean values:

$$\textit{If } \sum_x \phi(x)h^2 = 0, \textit{ then } \sum_k (P\phi)(X_k)V[X_k] = 0 .$$

I0R: R preserves zero mean values:

$$\textit{If } \sum_k \Phi(X_k)V[X_k] = 0, \textit{ then } \sum_x (R\Phi)(x)h^2 = 0 .$$

Here $\sum_x$ stands for $\sum_{x \in \Gamma^h \cap [0,1]^2}$. I0P and I0R state that the compatibility conditions are preserved if c=0. For our two-level algorithm, I0R is particularly useful, since it ensures solvability of the problems on Γ^h if c=0.

We also consider the following related properties.

ICP: P(1) has the discrete integral 1:

$$\sum_k (P1)(\underline{X}_k)V[\underline{X}_k] = 1 .$$

ICR: R(1) has the discrete integral 1:

$$\sum_{\underline{x}} (R1)(\underline{x})h^2 = 1 .$$

IP: P preserves discrete integrals:

$$\sum_k (P\phi)(\underline{X}_k)V[\underline{X}_k] = \sum_{\underline{x}} \phi(\underline{x})h^2 .$$

IR: R preserves discrete integrals:

$$\sum_{\underline{x}} (R\Phi)(\underline{x})h^2 = \sum_k \Phi(\underline{X}_k)V[\underline{X}_k].$$

The following figure shows trivial implications.

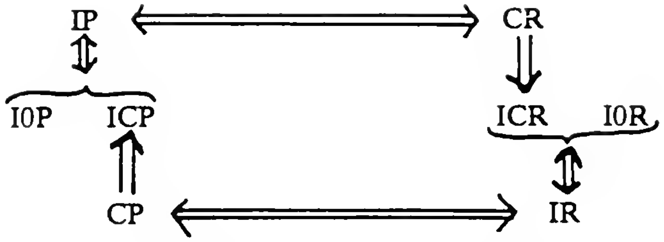
$$\begin{array}{ccc} & \mathrm{IP} & \\ & \Updownarrow & \\ \mathrm{I0P} & \overbrace{\qquad\qquad}^{} & \mathrm{ICP} \\ & & \Uparrow \\ & & \mathrm{CP} \end{array} \qquad\qquad \begin{array}{ccc} \mathrm{CR} & & \\ \Downarrow & & \\ \mathrm{ICR} & \underbrace{\qquad\qquad}_{} & \mathrm{I0R} \\ & \Updownarrow & \\ & \mathrm{IR} & \end{array}$$

We assume now that $P=R^*$, where R^* is the adjoint of R with respect to the discrete L^2-products on Γ^h and on S_N, see (3.19). This means

$$(P\phi)(\underline{X}_k) = \sum_{\underline{x}} c_{k\underline{x}}\, \phi(\underline{x})h^2 \text{ and} \tag{4.7}$$

$$(R\Phi)(\underline{x}) = \sum_k c_{k\underline{x}}\, \Phi(\underline{X}_k)V[\underline{X}_k], \tag{4.8}$$

where the coefficients $c_{k\underline{x}}$ are the same in both equations. Then we obtain the additional implications shown in the following figure.



We focus our attention on operators defined as convolutions of the form

$$(P\phi)(X) := \sum_{x\in\Gamma^h} \phi(x)\delta_h(X - x)h^2, \tag{4.9}$$

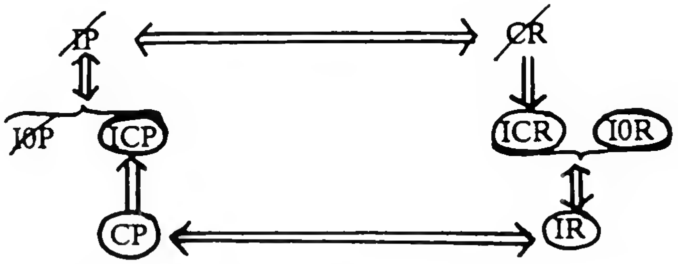
$$(R\Phi)(x) := \sum_{X\in S_N} \Phi(X)\delta_h(x - X)V[X], \tag{4.10}$$

where δ_h is a spread-out model of the delta distribution with support of linear size $O(h)$. (4.9) and (4.10) imply $P=R^*$. It is clear that CR cannot be satisfied by an operator R of this form, for it is possible that no point in S_N lies close to $x\in\Gamma^h$, in which case $R\Phi(x)$ is 0, no matter what Φ is. CP, on the other hand, is easily satisfied, for example with

$$\delta_h(x) = \delta_h(x_1)\delta_h(x_2), \ \ with \tag{4.11}$$

$$\delta_h(x) = \begin{cases} \frac{1}{h}(1 - \frac{|x|}{h}) \text{ if } |x|\le h \ , \\ 0 \ otherwise. \end{cases} \tag{4.12}$$

Notice that (4.9),(4.11),(4.12) is just piecewise bilinear interpolation. The following figure shows which of the mentioned properties the resulting operators R and P have and which ones they do not have. Our considerations show that no operator of the form (4.9),(4.10) can have the missing properties.



Eqs. (4.9) - (4.12) specify our choice of P and R. It remains to choose a relaxation scheme. The relaxation schemes which we shall describe are applied to equations of the

form

$$\sum_{Y \neq X_k} A[X_k, Y] \frac{\phi(Y) - \phi(X_k)}{\|Y - X_k\|} + V[X_k] c \phi(X_k) = V[X_k] f(X_k),$$

i.e. we multiply the k-th equation by $V[X_k]$. This has the effect that the matrix is positive definite and symmetric with respect to the euclidean inner product. It is crucial to compensate for this multiplication in some way when transferring residuals to the regular grid. The same issue occurs when solving a second order equation on a regular grid using a multigrid method. In that case, the equations are often multiplied by the squares of the meshwidths on all grids. A common way of compensating for this scaling is to multiply the restriction operator from a grid with meshwidth h_1 to a coarser grid with meshwidth h_2 by $\frac{h_2^2}{h_1^2}$. Similarly, we multiply the equations on the regular grid by h^2 and multiply the matrix R from the right by the diagonal matrix whose k-th diagonal entry is $\frac{h^2}{V[X_k]}$. This means that we remove the factors $V[X_k]$ before computing the residuals.

Using Gauss-Seidel relaxation, we obtain a performance which is disappointing in comparison with many multigrid methods on regular grids. The following table shows the reduction of the discrete L^2-norms of the residual. The order in which the unknowns are relaxed is determined by the numbering of the grid points, which is random. In all cases, the continuous problem being solved has the solution

$$\phi(x) = sin(2\pi(x_1 - 2x_2)) . \tag{4.13}$$

Table 4.1. Residual reduction obtained with two-level method after 1, 3 and 5 iterations, Helmholtz equation, c=10, $\nu_1=2$, $\nu_2=1$, Gauss-Seidel relaxation. 10 random, uniformly distributed grids for each N. Best results, worst results and geometric means over all 10 experiments.

N	worst case	best case	average
	0.15E+00	0.73E-01	0.10E+00
100	0.12E-01	0.17E-03	0.12E-02
	0.96E-03	0.91E-06	0.22E-04
	0.28E+00	0.16E+00	0.22E+00
400	0.15E-01	0.14E-02	0.67E-02
	0.15E-02	0.32E-04	0.38E-03
	0.30E+00	0.21E+00	0.25E+00
1600	0.22E-01	0.42E-02	0.98E-02
	0.43E-02	0.34E-03	0.11E-02

The results can be improved by a modification of the relaxation method. This modification is an application of a general rule due to Brandt (1981) on block relaxation in multigrid methods. The improved relaxation scheme is defined as follows. Introduce the notation

$$V[\mathcal{X}_k](L\phi)(\mathcal{X}_k) =: \sum_{\mathcal{Y}} L[\mathcal{X}_k,\mathcal{Y}]\phi(\mathcal{Y}) \,. \tag{4.14}$$

When relaxing $\mathcal{X}_k$, determine the set of all neighbors $\mathcal{Y}$ of $\mathcal{X}_k$ for which

$$L[\mathcal{X}_k,\mathcal{Y}] \;\geq\; \delta(-L[\mathcal{X}_k,\mathcal{X}_k] + cV[\mathcal{X}_k]), \tag{4.15}$$

where $\delta \in [0;1]$ remains to be chosen. Change the values in all these neighbors and in $\mathcal{X}_k$ simultaneously so that their equations become satisfied. As a result, some of the $\phi(\mathcal{X}_k)$ may be modified more than once during one relaxation sweep. We note that our

criterion (4.15) for relaxing points as a block is not equivalent to an analogous criterion based on the distances between the grid points. We have conducted experiments which indicate that such a criterion leads to a less efficient algorithm than ours.

If δ is at least 0.5 and $c \geq 0$, the blocks which are relaxed simultaneously are at most of size 2. To see this, recall that the diagonal elements of L are negative, that the off-diagonal elements are positive or zero with at least 3 positive elements in each row, and that the row sums of L are zero. Thus there is at most one neighbor $\underline{Y}$ of any point $\underline{X}_k$ for which (4.15) is satisfied when $\delta \geq 0.5$. With $\delta = 1.0$, we obtain the usual Gauss-Seidel iteration, by a similar argument. We choose $\delta = 0.5$.

We repeat the above experiments with the modified relaxation scheme.

Table 4.2. Experiments as in table 4.1, with modified relaxation scheme.

N	worst case	best case	average
100	0.77E-01 0.30E-03 0.17E-05	0.28E-01 0.22E-04 0.31E-06	0.48E-01 0.72E-04 0.65E-06
400	0.14E+00 0.16E-02 0.31E-04	0.94E-01 0.14E-03 0.10E-05	0.11E+00 0.40E-03 0.40E-05
1600	0.16E+00 0.54E-03 0.17E-04	0.13E+00 0.27E-03 0.58E-05	0.14E+00 0.38E-03 0.78E-05

The results for c=0.0 are hardly different. Here we project the discrete right-hand side onto its constant-free part to ensure that there is a solution. It is unnecessary to impose any condition such as a prescribed mean or point value to ensure uniqueness of the constant component in the computed solution. The non-uniqueness of the solution

may simply be ignored.

Table 4.3. Experiments as in table 4.1, with modified relaxation scheme, c=0.0

N	worst case	best case	average
100	0.79E-01 0.40E-03 0.26E-05	0.32E-01 0.36E-04 0.30E-06	0.53E-01 0.10E-03 0.65E-06
400	0.14E+00 0.17E-02 0.33E-04	0.98E-01 0.17E-03 0.13E-05	0.12E+00 0.46E-03 0.42E-05
1600	0.16E+00 0.56E-03 0.20E-04	0.13E+00 0.28E-03 0.54E-05	0.15E+00 0.39E-03 0.78E-05

We still obtain convergence factors which increase with growing N. Good multigrid methods have convergence factors which are bounded independent of the number of unknowns, with a bound far below 1; see Stüben and Trottenberg (1982). We do not know whether our method has such a property. It is, in any case, satisfactory when used within our fractional step method, see section 6.

5. Projection of a vector field onto the kernel of the discrete divergence operator

The last tool needed for our fractional step method is an algorithm which projects a given vector field $\underline{u}$ on S_N orthogonally onto the kernel of the discrete divergence operator D. This means that we want to find $\mathbf{P}\underline{u}$ and q such that

$$\underline{u} = \mathbf{P}\underline{u} + \underline{G}q \quad \text{and} \tag{5.1}$$

$$D\mathbf{P}\underline{u} = 0 . \tag{5.2}$$

For this purpose, we have to solve

$$D\underline{G}q = D\underline{u} . \tag{5.3}$$

$D\underline{G}$ is singular, but $D\underline{u}$ satisfies the compatibility condition, i.e. $D\underline{u}$ is orthogonal to the kernel of $D\underline{G}$ in the inner product (3.19), and $\underline{G}q$ is unique.

On a square grid with meshwidth h, $D\underline{G}$ is the standard 5-point discretization of the Laplace operator with meshwidth 2h. Simple relaxation schemes have no smoothing effect for this discretization, as can immediately be verified by Fourier analysis. We ignore the fact that the system can be decoupled into 4 smaller systems which can be solved easily, since this will not be the case on irregular grids. The difficulty is related to the fact that there are highly oscillatory grid functions in the kernel of the finite difference operator, and such components are not affected at all by relaxation methods. By a continuity argument, the error in a nearby Fourier mode is damped slowly. Problems of this kind have been investigated much more generally by Brandt and Dinar (1979).

One might consider solving (5.3) by a conjugate gradient iteration with preconditioning. On a square mesh, it is useless to precondition $D\underline{G}$ with L. This can immediately be shown by discrete Fourier analysis. Even if a good preconditioner for $D\underline{G}$ could be found, the fact that we do not know much about the kernel of $D\underline{G}$, i.e. the kernel of $\underline{G}$, would present a problem. A non-trivial null-space does not affect the performance of the conjugate gradient method in exact arithmetic. In floating point arithmetic, however, the method often does not converge when applied to a positive semidefinite matrix

with a non-trivial null-space unless the computed residuals are projected into the orthogonal complement of the null-space.

We therefore replace the discrete projection operator

$$I - \underset{\sim}{G}(D\underset{\sim}{G})^{-1}D \qquad (5.4)$$

by

$$I - \underset{\sim}{G}L^{-1}D \; . \qquad (5.5)$$

$D\underset{\sim}{G}$ and L are singular. Nevertheless $\underset{\sim}{G}(D\underset{\sim}{G})^{-1}D$ and $\underset{\sim}{G}L^{-1}D$ have obvious well-defined meanings.

(5.5) is not a projection operator. In fact, the discrete L^2-norm

$$\|I - \underset{\sim}{G}L^{-1}D\| = \rho(I - \underset{\sim}{G}L^{-1}D) \qquad (5.6)$$

is often, i.e. on many fluid marker configurations, larger than 1. The eigenvalues with largest absolute value must clearly be smaller than -1 if that is the case.

Proposition 5.1: If all eigenvalues of $I - \underset{\sim}{G}L^{-1}D$ lie in $(-1;1]$, then

$$\lim_{j\to\infty} (I - \underset{\sim}{G}L^{-1}D)^j = I - \underset{\sim}{G}(D\underset{\sim}{G})^{-1}D \; . \qquad (5.7)$$

Proof:

$\underset{\sim}{v} := \lim_{j\to\infty} (I - \underset{\sim}{G}L^{-1}D)^j(\underset{\sim}{u})$ differs from $\underset{\sim}{u}$ only by a discrete gradient. Therefore it suffices to prove that $D\underset{\sim}{v}=0$. Clearly, $(I-\underset{\sim}{G}L^{-1}D)\underset{\sim}{v} = \underset{\sim}{v}$, therefore $\underset{\sim}{G}L^{-1}D\underset{\sim}{v}=0$. Using $\underset{\sim}{G}=-D^*$, we conclude that $L^{-1}D\underset{\sim}{v}=0$, therefore $D\underset{\sim}{v}=0$. As before, L^{-1} has an obvious meaning. ∎

We give a different interpretation of $(I-\underset{\sim}{G}L^{-1}D)^j\underset{\sim}{u}$. To solve (5.3), solve $Lq_1=D\underset{\sim}{u}$ first and perform $j-1$ defect correction iteration steps (see Stetter, 1978), using L on the left-hand side and $D\underset{\sim}{G}$ on the right-hand side, obtaining an approximation q_j. Then

$$\underset{\sim}{u} - \underset{\sim}{G}q_j = (I-\underset{\sim}{G}L^{-1}D)^j\underset{\sim}{u} \; .$$

Since (5.5) is a discretization of the continuous projection operator, it is to be expected that at least those eigenvalues which correspond to smooth eigenfunctions are larger than

-1. This motivates the following modification of (5.5):

$$I - \underline{G}(I + \omega L)^k L^{-1} D \ , \tag{5.8}$$

with $\omega \lessapprox 1/\rho(L)$ but $\omega \leq 1/\rho(L)$, k integer. In our experiments, we always take

$$\omega := \frac{1}{\bar{\rho}} \ , \tag{5.9}$$

where $\bar{\rho}$ is the maximum row sum norm of the matrix L, i.e.

$$\omega = \left(\max_k \frac{2}{V[\underline{X}_k]} \sum_{\underline{Y} \neq \underline{X}_k} \frac{A[\underline{X}_k, \underline{Y}]}{||\underline{Y} - \underline{X}_k||} \right)^{-1} . \tag{5.10}$$

We show that ω is at most of order $O(h^2)$. Using the estimates

$$||\underline{X} - \underline{Y}|| \leq 2h \tag{5.11}$$

$$\sum_{\underline{Y} \neq \underline{X}} A[\underline{X}, \underline{Y}] \geq 2\sqrt{\pi}\sqrt{V[\underline{X}]} \tag{5.12}$$

in (5.10), we obtain

$$\omega \leq \frac{h \min \sqrt{V[\underline{X}_k]}}{2\sqrt{\pi}} . \tag{5.13}$$

$I + \omega L$ is therefore pointwise consistent with the identity, since $L\phi$ is bounded for $h \to 0$ if ϕ is a fixed smooth periodic function; see proposition 3.4.

For fixed k, (5.8) can still be considered a discretization of the continuous projection operator.

Proposition 5.2: For sufficiently large k, the powers of (5.8) converge to the exact discrete projection operator $I - \underline{G}(D\underline{G})^{-1}D$.

Proof:

The proof that the limit, if it exists, is $I - \underline{G}(D\underline{G})^{-1}D$ is a duplicate of the proof of (5.7). To see that the limit of the powers of (5.8) exists, notice that $((I - \underline{G}(I + \omega L)^k L^{-1} D)$ is symmetric in the discrete L^2-product (3.19). We show that the Rayleigh quotient

$$\frac{((I-\underline{G}(I+\omega L)^k L^{-1}D)\underline{u},\underline{u})}{(\underline{u},\underline{u})} \tag{5.14}$$

lies in (-1;1] for large k. For this purpose, we rewrite (5.14) as

$$1 + \frac{((I+\omega L)^k L^{-1}D\underline{u},D\underline{u})}{(\underline{u},\underline{u})}. \tag{5.15}$$

Since $(I+\omega L)^k L^{-1}$ is symmetric and negative semidefinite with respect to (3.19), (5.15) is ≤ 1. For large k, (5.15) is >-1, because those eigenvalues of $(I+\omega L)^k$ which are different from the simple eigenvalue 1 tend to 0 for $k\to\infty$. ∎

This leads to the following algorithm, which chooses the smallest possible k itself:

Algorithm 5.1:

Given $\underline{u}$, generate $\underline{u}^0$, $\underline{u}^1$, $\underline{u}^2,\cdots$ *with* $\underline{u}^j \to \mathbf{P}\underline{u}$ as follows:

$\underline{u}^0 := \underline{u}$.

Given $\underline{u}^j$, define

$k := 0;\;\; \underline{u}^{j,0} := \underline{u}^j,\; q^{j,1} := L^{-1}D\underline{u}^{j,0};\;\; \underline{u}^{j,1} := \underline{u}^{j,0} - \underline{G}q^{j,1}.$

While $\|\,\underline{u}^{j,k+1}\,\| > \|\underline{u}^j\|$:

  $k := k+1;\;\; q^{j,k+1} := \omega L q^{j,k};\;\; \underline{u}^{j,k+1} := \underline{u}^{j,k} - \underline{G}q^{j,k+1}.$

$\underline{u}^{j+1} := \underline{u}^{j,k+1}.$

In our experiments, this algorithm usually requires k=0 when computing $\underline{u}^1$ from $\underline{u}^0$, k>0 for j>1, sometimes unfortunately k >> 0.

Notice that $\|\underline{u}^{j+1}\|\leq\|\underline{u}^j\|$. This property ensures the linear stability of the fractional step method introduced in section 6, even if only a few iterations of the projection algorithm 5.1 are used.

One can interpret the factor $(I+\omega L)^k$ in the following way. Application of $I+\omega L$ to a function q_0 is equivalent to performing one modified Jacobi relaxation sweep for the system $Lq=0$, with the initial guess q_0. We expect such a sweep to damp highly oscillatory components of q_0 and leave smooth components essentially unchanged. It might

therefore be useful to replace the factor $I + \omega L$ by one sweep of a more efficient smoothing method. Numerical results indicate, however, that the relaxation method used in section 4 is significantly inferior to modified Jacobi relaxation, i.e. to the smoothing factor $I + \omega L$, in the present context.

We give a numerical example. Consider

$$\underline{u}(\underline{x}) = (sin(2\pi x_1)\cos(2\pi x_2)\ ,\ 0)\ . \tag{5.16}$$

The divergence-free part of $\underline{u}$ is

$$0.5(sin(2\pi x_1)\cos(2\pi x_2)\ ,\ -\cos(2\pi x_1)\sin(2\pi x_2))\ . \tag{5.17}$$

We compute the divergence-free part of $\underline{u}$ numerically, compute the discrete L^2-norms of the discretization errors in the two components, and compute the square root of the sum of the squares of these two numbers. This results in the following table.

Table 5.1. Discretization errors obtained with algorithm 5.1 for example (5.16). 20 random, uniformly distributed grids for each N.

N	iteration	worst case	best case	N	iteration	worst case	best case
25	1	0.30	0.22	400	1	0.081	0.070
	2	0.30	0.20		2	0.078	0.061
	3	0.28	0.20		3	0.083	0.063
	4	0.29	0.19		4	0.081	0.064
	5	0.29	0.20		5	0.086	0.064
	10	0.29	0.20		10	0.084	0.062
	15	0.30	0.20		15	0.089	0.062
100	1	0.17	0.13	1600	1	0.041	0.035
	2	0.16	0.11		2	0.040	0.031
	3	0.16	0.11		3	*	0.033
	4	0.16	0.11		4	*	0.034
	5	0.16	0.11		5	*	0.035
	10	0.16	0.11		10	*	0.037
	15	0.16	0.11		15	*	0.037

"*" means that $k=50$ is not sufficient. In none of the experiments summarized in the table, k is larger than 0 in the first iteration or larger than 2 in the second iteration.

One iteration generates an approximation to the continuous solution which is as good as or even better than the approximation obtained after many iterations. What we have gained by replacing (5.5) with (5.8) are guaranteed stability and convergence to the solution of the discrete projection problem, not higher accuracy.

It is surprising that the truncation error does decrease with growing N here, even though D is not pointwise consistent with the divergence operator. We also observe that the variance of the truncation error seems to decrease with growing N. These two

statements are based on numerical results. We have not been able to prove them.

The operator D_2 introduced in section 3, i.e. the operator which is obtained by discretizing the divergence theorem on the Voronoi polygons in the most obvious way, appears to give far worse results here. We illustrate this by recomputing a part of the previous table, now using D_2 rather than $D=D_1$ and $\underline{G}_2$, the negative adjoint of D_2, rather than $\underline{G}=\underline{G}_1$.

Table 5.2. Experiments as in table 4.1, with $D,\underline{G}$ replaced by $D_2,\underline{G}_2$.

N	iteration	worst case	best case	N	iteration	worst case	best case
25	1	0.29	0.22	400	1	0.11	0.095
	2	0.27	0.18		2	0.098	0.085
100	1	0.16	0.12	1600	1	0.094	0.086
	2	0.14	0.11		2	0.083	0.076

6. A fractional step method for the Navier-Stokes equations

We consider the following fractional step method for the Navier-Stokes equations.

$$\begin{aligned} \frac{\tilde{\underline{u}}^{n+1} - \underline{u}^n}{\Delta t} - \nu L^n \tilde{\underline{u}}^{n+1} &= \underline{f}^{n+1} \\ \underline{u}^{n+1} &= \mathbf{P}^n \tilde{\underline{u}}^{n+1} \\ \underline{X}^{n+1} &= \underline{X}^n + \Delta t \underline{u}^{n+1} . \end{aligned} \tag{6.1}$$

Here L^n denotes the discrete Laplacian on S_N^n, the set S_N at time $n\Delta t$. The points in that set are the components of the vector $\underline{X}^n$. $\mathbf{P}^n$ is the orthogonal projection onto the kernel of D^n, where D^n is the discrete divergence operator on S_N^n. The pressure is treated in (6.1) as in the fractional step methods introduced and analyzed by Chorin (1968,1969) and Temam (1969). The nonlinear term is absent in this Lagrangian model.

Writing $\mathbf{P}^n \tilde{\underline{u}}^{n+1} =: \tilde{\underline{u}}^{n+1} - \underline{G}^n q^{n+1}$, we obtain

$$\begin{aligned} \frac{\underline{u}^{n+1} - \underline{u}^n}{\Delta t} - \nu L^n \tilde{\underline{u}}^{n+1} + \frac{\underline{G}^n q^{n+1}}{\Delta t} &= \underline{f}^{n+1} \quad \text{and} \\ D^n \underline{u}^{n+1} &= 0 . \end{aligned} \tag{6.2}$$

Therefore $p^{n+1} := q^{n+1}/\Delta t$ is an approximation of the pressure. The formulas (6.2) with $L^n \underline{u}^{n+1}$ instead of $L^n \tilde{\underline{u}}^{n+1}$, together with the third equation of (6.1), define the method introduced by Peskin (1979).

(6.1) is unconditionally linearly stable in the sense that

$$\|\underline{u}^{n+1}\|_{(n)} \leq \|\underline{u}^n\|_{(n)} + \Delta t \|\underline{f}^{n+1}\|_{(n)} \tag{6.3}$$

where $\|\cdot\cdot\|_{(n)}$ denotes the discrete L^2-norm on S_N^n. This follows from proposition 3.5. Using the projection algorithm introduced in section 5, stability is guaranteed even though we do no apply the operator $\mathbf{P}^n$ exactly.

We also consider the following modification of (6.1).

$$\frac{\tilde{\underline{u}}^{n+1} - \mathbf{P}^n \underline{u}^n}{\Delta t} - \nu L^n \tilde{\underline{u}}^{n+1} = \mathbf{P}^n \underline{f}^{n+1}$$

$$\underline{u}^{n+1} = \mathbf{P}^n \tilde{\underline{u}}^{n+1} \tag{6.1b}$$

$$\underline{X}^{n+1} = \underline{X}^n + \Delta t \underline{u}^{n+1} .$$

Notice that this method requires two, not three applications of $\mathbf{P}^n$ per time step.

Writing, as before, $\mathbf{P}^n \tilde{\underline{u}}^{n+1} =: \tilde{\underline{u}}^{n+1} - \underline{G}^n q^{n+1}$, and writing

$$\mathbf{P}^n(\underline{u}^n + \Delta t \underline{f}^{n+1}) =: \underline{u}^n + \Delta t \underline{f}^{n+1} - \underline{G}^n r^{n+1},$$

we obtain

$$\frac{\underline{u}^{n+1} - \underline{u}^n}{\Delta t} - \nu L^n \tilde{\underline{u}}^{n+1} + \underline{G}^n \left(\frac{q^{n+1} + r^{n+1}}{\Delta t}\right) = \underline{f}^{n+1} \quad \text{and} \tag{6.2b}$$

$$D^n \underline{u}^{n+1} = 0 .$$

Therefore $p^{n+1} := \dfrac{q^{n+1} + r^{n+1}}{\Delta t}$ is an approximation of the pressure.

Those of the following numerical results for which the method is not specified are obtained with method (6.1).

Test problem 1:

We construct an exterior force density such that the flow becomes

$$\begin{aligned} u_1(\underline{x},t) &= \sin(\tfrac{\pi}{2}t)\cos(2\pi x_1)\sin(2\pi x_2) \\ u_2(\underline{x},t) &= -\sin(\tfrac{\pi}{2}t)\sin(2\pi x_1)\cos(2\pi x_2) \\ p(\underline{x},t) &= \sin(\tfrac{\pi}{2}t)\cos(2\pi x_1)\cos(2\pi x_2). \end{aligned} \tag{6.4}$$

We measure the discrete L^2-norms of the errors in velocity and pressure at time 1. In every time step, we use only one step of algorithm 5.1. Initially, the $\underline{X}_j$ are at the positions $((k_1 + 0.5)h, (k_2 + 0.5)h)$, $0 \le k_1 \le \sqrt{N} - 1$, $0 \le k_2 \le \sqrt{N} - 1$. We expect that the solution of the scalar Helmholtz and Poisson problems up to rounding error accuracy is necessary only at time 0. At a later time step, the values from the previous time step should provide an excellent initial guess, and little work should be required to improve this guess such that the truncation error level is reached. We use only one two-level

cycle per scalar problem, with $\nu_1=2$, $\nu_2=1$. The following tables show results of our experiments. We use the abbreviations $h_N := \frac{1}{\sqrt{N}}$, $e_1 :=$ error in u_1, $e_2 :=$ error in u_2 and $e_p :=$ error in p. Since the average of the exact pressure is zero, we project the computed approximation for the pressure onto its constant free part with respect to the discrete L^2-product before measuring e_p.

Table 6.1. Errors in velocity and pressure, test problem 1, $\Delta t=h_N$, $\nu=1.0$, starting on square grids.

N	e_1	e_2	e_p
100	0.35	0.35	0.11
400	0.12	0.12	0.38
1600	0.063	0.064	0.33

From these results, it seems unlikely that the approximation for the pressure converges to the exact pressure at all in this case.

We repeat the experiments with lower viscosity.

Table 6.2. Errors in velocity and pressure, test problem 1, $\Delta t=h_N$, $\nu=0.1$, starting on square grids.

N	e_1	e_2	e_p
100	0.35	0.35	0.11
400	0.12	0.12	0.12
1600	0.064	0.065	0.082

The following table contains results obtained with random initial grids. As before, we use one two-level cycle per scalar problem. Notice that the errors do not differ very

much from those displayed in table 6.2.

Table 6.3. Errors in velocity and pressure, test problem 1, $\Delta t = h_N$, $\nu = 0.1$, starting on 20 uniformly distributed, random grids.

N	e_1			e_2			e_p		
	worst	best	average	worst	best	average	worst	best	average
100	0.51	0.31	0.38	0.48	0.29	0.36	0.37	0.20	0.27
400	0.17	0.13	0.15	0.18	0.13	0.15	0.14	0.086	0.12
1600	0.064	0.059	0.061	0.062	0.059	0.061	0.076	0.069	0.072

As in section 5, we observe that the variance of the error appears to decrease for increasing N.

We repeat some of the above experiments, now solving all scalar problems up to truncation error accuracy, but still applying only one step of algorithm 5.1. The results confirm that one two-level cycle per scalar problem is sufficient:

Table 6.4. Experiments as in table 6.2, solving all scalar problems up to rounding error accuracy.

N	e_1	e_2	e_p
100	0.35	0.35	0.11
400	0.12	0.12	0.12
1600	0.067	0.068	0.086

The slight difference between the errors in the two velocity components for N=1600 is caused by rounding. In exact arithmetic, $e_1 = e_2$.

Using smaller viscosity coefficients, for example $\nu = 10^{-4}$, we still obtain reasonable results for this problem. However, some of our results for test problem 4 below indicate that the method cannot generally be used for such viscosity coefficients.

Test problem 2:

We consider a shear flow:

$$\begin{aligned} u_1(\underline{x},t) &= \sin(2\pi x_2) \\ u_2(\underline{x},t) &= 0 \\ p(\underline{x},t) &= 0 \end{aligned} \tag{6.6}$$

Proposition 6.1: If the initial marker configuration is a rectangular grid with meshwidths Δx_1 and Δx_2,

$$\Delta x_2 \geq \frac{1}{2}\Delta x_1, \tag{6.7}$$

and if the initial velocity and the exterior force are independent of x_1 and parallel to the x_1-axis, then the discrete approximation for the second velocity component obtained by (6.1) or (6.1b) is 0.

Proof:

The statement would be obvious if $\mathbf{P}^n$ were replaced by the identity operator. Therefore, it suffices to prove the following statement.

Assume that

$$\{\underline{X}_1,\cdots,\underline{X}_N\} = \{((i+\frac{1}{2})\Delta x_1+\delta_j,(j+\frac{1}{2})\Delta x_2) : i=0,\cdots,\frac{1}{\Delta x_1}-1,\ j=0,\cdots,\frac{1}{\Delta x_2}-1\} \tag{6.8}$$

with numbers

$$\delta_j \in [-\frac{1}{2}\Delta x_1, \frac{1}{2}\Delta x_1] \tag{6.9}$$

independent of i, and assume that (6.7) holds. Consider a vector field

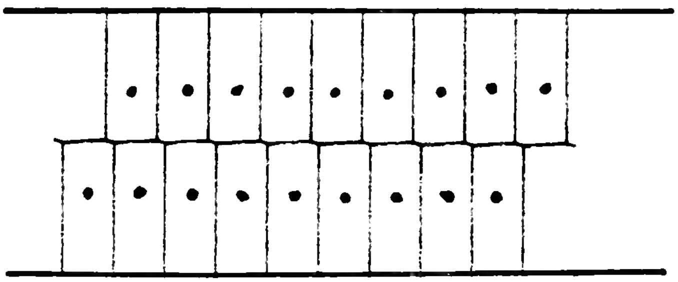
$$\underline{u}(\underline{x}) = \begin{pmatrix} u_1(x_2) \\ 0 \end{pmatrix}. \tag{6.10}$$

Then

$$Du(X_k)=0 \qquad (6.1\ )$$

for all k.

To prove this, consider two rows of fluid markers and the associated Voronoi diagram in an infinite (periodic) horizontal strip, as indicated in the following figure.



All Voronoi polygons have the same shape and therefore the same area.

Now we consider the case of more than two rows of markers. (6.7) ensures that the Voronoi polygons in row j+1 are not adjacent to the Voronoi polygons in row $j-1$. The Voronoi polygons in row j are similar to each other, and the total area which they occupy within one period is constant. Therefore each individual polygon has a constant area, which implies (6.11). ■

We start the calculation on random grids. We obtain the following results.

Table 6.5. Errors in velocity and pressure, test problem 2, $\Delta t = h_N$, $\nu = 1.0$, starting on 10 random, uniformly distributed grids.

N	e_1			e_2			e_p		
	worst	best	average	worst	best	average	worst	best	average
100	0.075	0.045	0.060	0.071	0.041	0.056	0.061	0.041	0.053
400	0.035	0.025	0.030	0.035	0.024	0.029	0.033	0.022	0.027
1600	0.016	0.014	0.015	0.015	0.013	0.014	0.015	0.013	0.014

Test problem 3:

The solutions $\underline{u}$ of problems 1 and 2 have the special property that $\underline{u} \cdot \underline{\nabla} \underline{u}$ is a gradient. Therefore we test the method on another problem with a simple prescribed solution which does not have this property:

$$\begin{aligned} u_1(\underline{x},t) &= \sin^2(2\pi x_2) \\ u_2(\underline{x},t) &= \cos(2\pi x_1) \\ p(\underline{x},t) &= \sin^2(2\pi x_1) + \sin^2(2\pi x_2) \end{aligned} \tag{6.12}$$

At time 0, we solve the scalar problems up to rounding error. At later time steps, we solve them approximately, using one two-level cycle. We obtain the following results.

Table 6.6. Errors in velocity and pressure, test problem 3, $\Delta t = h_N$, $\nu = 0.1$, starting on square grids.

N	e_1	e_2	e_p
100	0.31	0.25	0.47
400	0.14	0.12	0.27
1600	0.070	0.058	0.16

Test problem 4:

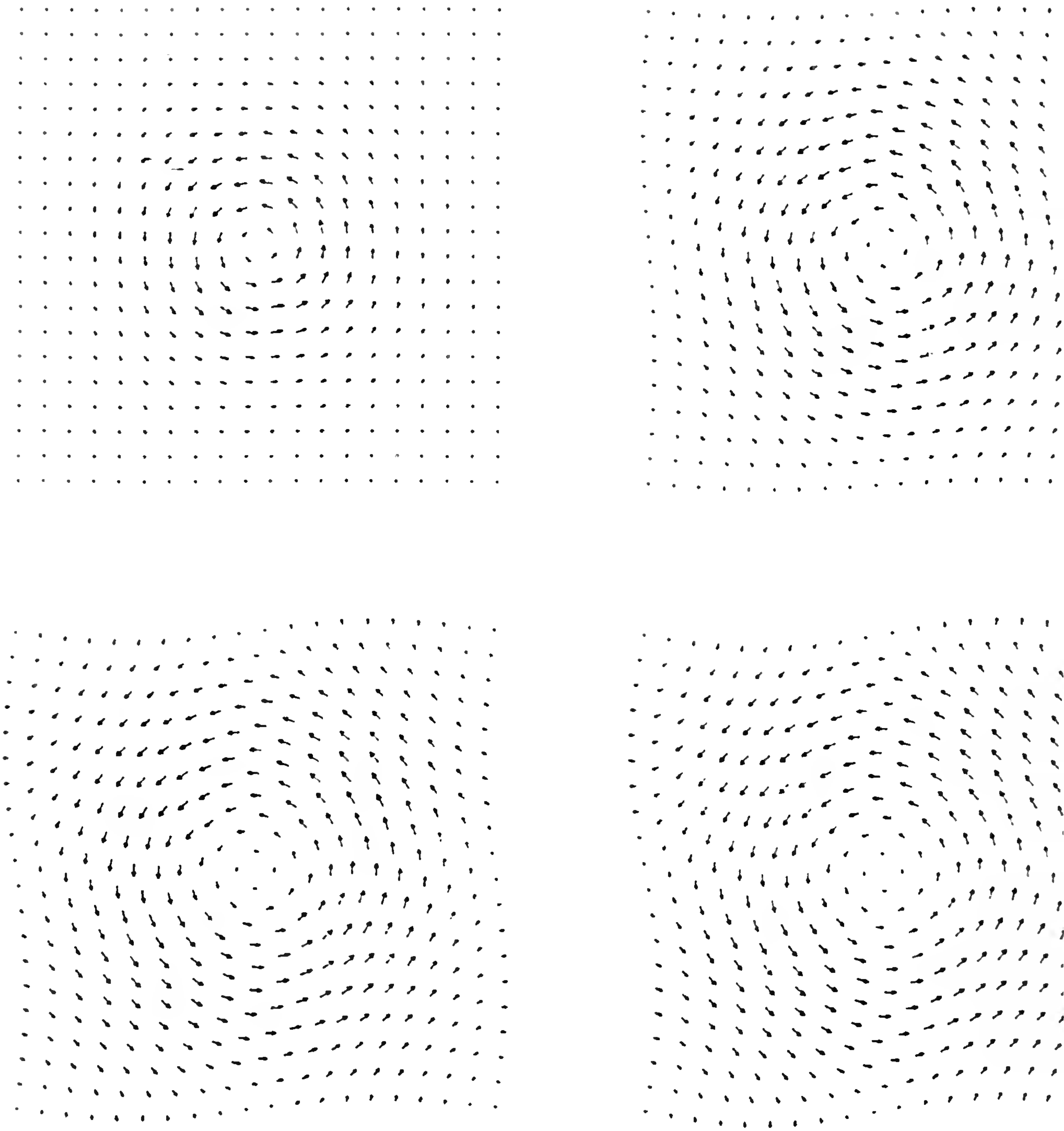
We consider an initial vortex blob centered at (0.5,0.5) and compute the flow which results in the absence of any exterior force. To obtain the initial velocity field, we solve the discrete Poisson problem with periodic boundary conditions and the right-hand side

$$\begin{cases} \frac{1}{0.16}(1+\cos(\frac{\pi(x_1-0.5)}{0.2}))(1+\cos(\frac{\pi(x_2-0.5)}{0.2})) + c & \text{if } |x_1-0.5|\le 0.2 \text{ and } |x_2-0.5|\le 0.2 \\ c & \text{otherwise} \end{cases} \qquad (6.13),$$

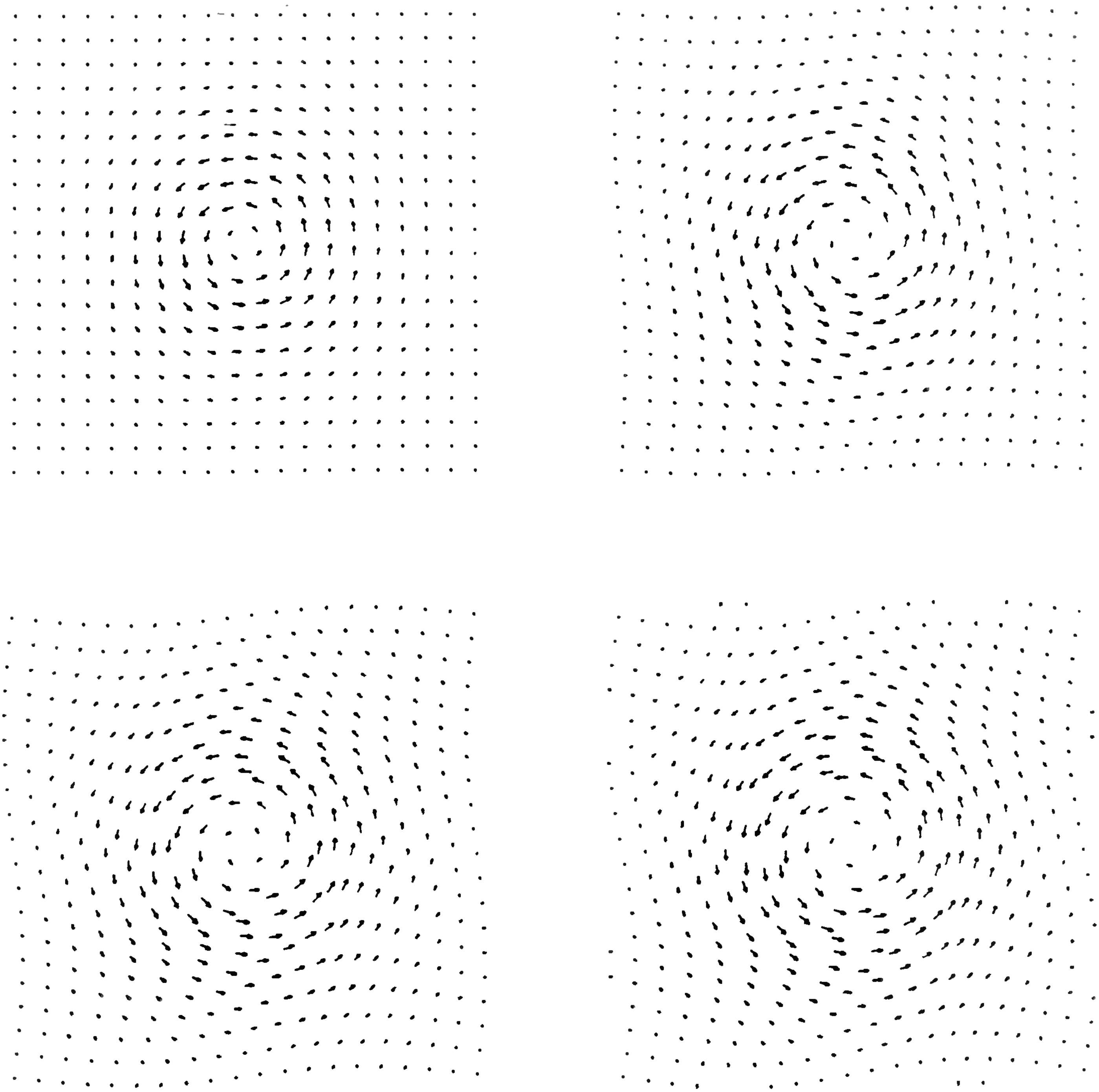
where the constant c is chosen such that the discrete compatibility condition is satisfied. From the resulting discrete stream function ψ^h, we obtain the velocity using central differencing.

Our numerical results for this example show that the centrifugal force tends to push the markers away from the center, leaving a region in which there are no markers at all. To counteract this tendency, we use the modified method (6.1b), in which the velocity field is projected onto its divergence free component not only before but also after moving the markers. Again we use only one step of algorithm 5.1 in each projection.

The following figures show computed flow fields for N=400, $\nu=0.1$, $\Delta t=0.025$, at time t = 0.0, 0.1, 0.2, 0.3. Δt was 0.025 here. Due to viscosity, the velocity decreases rapidly. The maximum computed speed is 0.89 at time 0.0 and 0.14 at time 0.3. This decay is not visible in our plots since we normalize the lengths of the arrows such that the maximum length is the same for each plot. The plots obtained at later time do not differ much from the one at time 0.3.



We repeat this computation with $\nu=0.01$ and obtain the following fields at the same times:



We perform the following convergence test. We perform the computations with N=100,400 and 1600, each time starting on

$$\{(k_1h_N, k_2h_N) : 1 \le k_1, k_2 \le \sqrt{N}\}.$$

Notice that the initial marker sets for N=400, N=1600 contain the initial marker set for N=100 in this case. Let $X_1^{100}, \cdots, X_{100}^{100}$ be the marker positions for N=100 at time 0.3.

Let X_i^{400} and X_i^{1600} be the positions of the same markers at time 0.3 computed with N=400 and N=1600. We measure

$$e_{100,400} := \frac{\sum_{i=1}^{100} ||X_i^{100} - X_i^{400}||}{100}$$

and

$$e_{400,1600} := \frac{\sum_{i=1}^{100} ||X_i^{400} - X_i^{1600}||}{100}.$$

For a first order convergent method, one would expect

$$e_{400,1600} \approx \frac{e_{100,400}}{2}.$$

The computed values are even slightly more favorable:

Table 6.7. Test problem 4, $e_{100,400}$ and $e_{400,1600}$, $\Delta t = \frac{h_N}{2}$

ν	$e_{100,400}$	$e_{400,1600}$
0.1	0.0090	0.0040
0.01	0.065	0.018
0.001	0.12	0.043
0.0001	0.17	0.086

Thus convergence for $N \to \infty$ is apparently obtained even for very small viscosity coefficients. Notice, however, that the size of the error increases substantially for $\nu \to 0$ when N is fixed.

7. Elastic boundaries immersed in the fluid

In this section, we report on some preliminary numerical experiments with immersed boundaries. The mathematical model and the numerical method which we use here are due to Peskin (1977); see also Peskin and McQueen (1980).

We first present the mathematical model. Consider flow in a region Ω with an elastic, massless boundary immersed in the fluid. Let $\underline{X}(s,t)$ be a representation of this boundary at time t such that, for fixed s, $\underline{X}(s,t)$ is the trajectory of a fixed boundary point.

The boundary exerts a force on the fluid which can be determined in the following way. Consider an arc $s\in(a;b)$ of the boundary. The mass of this arc is zero, therefore its momentum and the rate of change of its momentum are zero. The sum of the forces exerted on it must therefore be zero. The fluid exerts a force on the arc, which we assume to be of the form

$$-\int_a^b \underline{f}(s,t)ds. \tag{7.1}$$

Therefore, the force exerted by the arc on the fluid is

$$\int_a^b \underline{f}(s,t)ds \tag{7.2}$$

A second force is exerted on the arc by the boundary itself. It is equal to

$$T\underline{\tau}|_a^b, \tag{7.3}$$

where T is the tension, defined by

$$T := S(\|\frac{\partial \underline{X}(s,t)}{\partial s}\| - 1)\ , \quad \text{and} \tag{7.4}$$

$$\underline{\tau} := \frac{\frac{\partial \underline{X}(s,t)}{\partial s}}{\|\frac{\partial \underline{X}(s,t)}{\partial s}\|}. \tag{7.5}$$

S is a material constant, a stiffness coefficient. We conclude

$$\int_a^b \underline{f}(s,t)ds = T\underline{\tau}|_a^b. \tag{7.6}$$

Formally, our mathematical problem is then

$$
\begin{aligned}
\underline{u}_t + (\underline{u}\cdot\underline{\nabla})\underline{u} - \nu\Delta\underline{u} + \underline{\nabla}p &= \underline{F}(\underline{x},t) \\
\underline{\nabla}\cdot\underline{u} &= 0 \\
\frac{\partial\underline{X}(s,t)}{\partial t} &= \underline{u}(\underline{X}(s,t),t) \\
\underline{u}(\underline{x},0) &= \underline{0} \\
\underline{X}(s,0) &= \underline{X}_0(s),
\end{aligned}
\tag{7.7}
$$

where $\underline{F}(\underline{x},t)$ is, for fixed t, the distribution whose action on vector valued test functions is defined by

$$(\underline{F},\underline{w}) := \int_I \frac{\partial(T\underline{\tau})}{\partial s}\cdot\underline{w}(\underline{X}(s,t))ds. \tag{7.8}$$

We have assumed here that the density ρ equals 1. A change in ρ has the effect of changing S in this model. Only $\frac{S}{\rho}$ is relevant.

Next we describe the numerical method. We consider a case in which the boundary is a closed simple curve. It is represented by M fluid markers $\underline{X}_1,\cdots,\underline{X}_M$. The remaining $N-M$ markers represent the fluid. We number the particles such that the j-th marker is next to the (j−1)-st and (j+1)-st markers in the circular chain of markers representing the boundary. In this section, index calculations are carried out modulo M.

We number the boundary markers first for notational convenience only. In our code, the boundary markers are actually numbered last. This has slight advantages for the construction of Voronoi diagrams, since the Voronoi diagrams associated with the boundary markers only are degenerate.

For $j\in\{1;\cdots;M\}$, the force on $P[\underline{X}_j]$ is defined by

$$S\left(\frac{\|\underline{X}_{j+1}-\underline{X}_j\|}{\Delta s}-1\right)\frac{\underline{X}_{j+1}-\underline{X}_j}{\|\underline{X}_{j+1}-\underline{X}_j\|} + S\left(\frac{\|\underline{X}_j-\underline{X}_{j-1}\|}{\Delta s}-1\right)\frac{\underline{X}_{j-1}-\underline{X}_j}{\|\underline{X}_{j-1}-\underline{X}_j\|}, \tag{7.9}$$

which discretizes $T\underline{\tau}|_a^b$. Δs is an input parameter, the resting length of the springs which form the boundary. Δs determines the possible resting positions of the boundary. A corresponding input to the continuous model is the parametrization of the initial curve

$X_0(s)$.

Following Peskin (1977), we compute the forces exerted by the boundary on the fluid in the following implicit way. Given marker positions X_j^n and velocities U_j^n at time $n\Delta t$ (j=1,...,N), we first compute approximate new positions $X_1^*,\cdots,X_N^*$ of the boundary markers. These approximate new positions are only used to compute the force exerted by the boundary on the fluid. The actual new positions are then obtained by advancing X_j,U_j by one time step in exactly the same way as in section 6, using the computed force exerted by the boundary as the right-hand side.

We compute the auxiliary positions $X_1^*,\cdots,X_N^*$ from

$$X_j^* = X_j^n + \Delta t U_j^n +$$

$$\frac{(\Delta t)^2}{V[X_j^n]}\left[S\left(\frac{\|X_{j+1}^*-X_j^*\|}{\Delta s}-1\right)\frac{X_{j+1}^*-X_j^*}{\|X_{j+1}^*-X_j^*\|} + S\left(\frac{\|X_j^*-X_{j-1}^*\|}{\Delta s}-1\right)\frac{X_{j-1}^*-X_j^*}{\|X_{j-1}^*-X_j^*\|}\right]. \quad (7.10)$$

As in the continuous case, the density is assumed to be 1 here, and a change in density can be interpreted as a change in the stiffness S.

We introduce the following simplifying notation:

$$\xi_j := \left(\frac{(\Delta t)^2 S}{V[X_j^n]\Delta s}\right)^{\frac{1}{2}}, \quad (7.11)$$

$$X := (X_1,\cdots,X_N), \quad (7.12)$$

$$E(X) := \sum_j \left[\frac{1}{2}(X_{j+1}-X_j)^2 - \Delta s\|X_{j+1}-X_j\|\right], \quad (7.13)$$

$$X_j^0 := X_j^n + \Delta t U_j^n. \quad (7.14)$$

(7.10) can then be rewritten as

$$X_j^* = X_j^0 - \xi_j^2 \nabla E(X^*). \quad (7.15)$$

As described so far, the discrete model describes the boundary as a circular chain of springs which have a resting length Δs and resist compression as well as extension. Peskin (1977) showed that the solution of (7.15) becomes much simpler if the resistance against compression is dropped. This means that we modify the definition of E in the

following way:

$$E(X) := \sum_j \begin{cases} \frac{1}{2}(\|X_{j+1} - X_j\| - \Delta s)^2 & \text{if } \|X_{j+1} - X_j\| \geq \Delta s \\ 0 & \text{otherwise.} \end{cases} \tag{7.16}$$

E is then once but not twice continuously differentiable.

With this modification, the solvability of (7.15) can be established and a solution can be found in the following way. Let

$$Y^0 := (\xi_1^{-1} X_1^0, \cdots, \xi_M^{-1} X_M^0) \tag{7.17}$$

and consider

$$\tilde{E}(Y) := \frac{1}{2}\|Y - Y^0\|^2 + E(\xi_1 Y_1, \cdots, \xi_M Y_M), \tag{7.18}$$

a function of $Y = (Y_1, \cdots, Y_M)$. $\tilde{E}$ has at least one local minimum, since $E \geq 0$. Let Y be a local minimum of $\tilde{E}$. Then

$$Y_j = Y_j^0 - \xi_j (\nabla E)(\xi_1 Y_1, \cdots, \xi_M Y_M). \tag{7.19}$$

Let

$$X_j := \xi_j Y_j, \tag{7.20}$$

then

$$X_j = X_j^0 - \xi_j^2 \nabla E(X_1, \cdots X_M). \tag{7.21}$$

An analogous change of coordinates can be used, more generally, for systems of the form

$$X = X_0 - A \nabla E(X) \tag{7.22}$$

with a positive definite matrix A and a function E bounded from below.

We introduce the notation

$$\delta_j := \xi_{j+1} Y_{j+1} - \xi_j Y_j. \tag{7.23}$$

Then $\tilde{E}(Y)$ can be written as

$$\frac{1}{2}\sum_{j=1}^{M}(X_j-X_j^0)^2 + \frac{1}{2}\sum_{j=1}^{M}\begin{cases}(\|\delta_j\|-\Delta s)^2 & \text{if } \|\delta_j\|\geq\Delta s\\ 0 & \text{otherwise.}\end{cases} \tag{7.24}$$

We use Newton's method with line search to minimize (7.24). For this, we compute the gradient and the Hessian matrix of (7.24). The gradient is

$$X_j-X_j^0+\xi_j\left[-\max(\|\delta_j\|-\Delta s,0)\frac{\delta_j}{\|\delta_j\|}+\max(\|\delta_{j-1}\|-\Delta s,0)\frac{\delta_{j-1}}{\|\delta_{j-1}\|}\right]. \tag{7.25}$$

The Hessian matrix is

$$I + (\xi_i\xi_j C_{i,j}),\ i=1,\ldots,M,\ j=1,\ldots,M, \tag{7.26}$$

where the $C_{i,j}$ are 2×2 blocks given by the following formulae.

$$C_{i,j} = 0 \ \text{ if } \ |(i-j)(\text{mod M})|\geq 2 \tag{7.27}$$

$$C_{i,i} = -C_{i,i-1} - C_{i,i+1} \tag{7.28}$$

$$C_{i+1,i} = C_{i,i+1} \tag{7.29}$$

$$C_{i,i+1} = -I+\left[\frac{\Delta s}{\|\delta_i\|^3}\begin{pmatrix}\delta_{i,2}^2 & -\delta_{i,1}\delta_{i,2}\\ -\delta_{i,1}\delta_{i,2} & \delta_{i,1}^2\end{pmatrix}\right], \tag{7.30}$$

with the term $[\cdots]$ in (7.30) replaced by 0 if $\|\delta_i\|<\Delta s$. Notice that all $C_{i,j}$ are symmetric.

Peskin (1977) proved that the Hessian is positive definite. We present this proof in our case.

Proposition 7.1: The matrix (7.26) is positive definite.

Proof:

First, we rewrite the Hessian as

$$I + \operatorname{diag}(\xi_i)\cdot \mathbf{C}\cdot \operatorname{diag}(\xi_i), \tag{7.31}$$

where

$$\mathbf{C} := (C_{i,j}). \tag{7.32}$$

It suffices to prove that **C** is positive semidefinite. We first show that the matrices $C_{l,l+1}$ are negative semidefinite. $C_{l,l+1}$ differs from $-I$ by a singular summand, so one of its eigenvalues is -1. The sum of the eigenvalues of $C_{l,l+1}$ is

$$\mathrm{trace}(C_{l,l+1}) = -2 + \begin{cases} \dfrac{\Delta s}{\|\delta_l\|} & \text{if } \Delta s \le \delta_l \\ -2 & \text{otherwise.} \end{cases} \tag{7.33}$$

Therefore the second eigenvalue must be nonpositive.

We conclude from this that **C** is positive semidefinite. Let $d_1, \cdots, d_M$ be vectors of length 2. We have to show that

$$\sum_l d_l^T \cdot C_{l,l} \cdot d_l + \sum_l d_l^T \cdot C_{l,l+1} \cdot d_{l+1} + \sum_l d_{l+1}^T \cdot C_{l,l+1} \cdot d_l \tag{7.34}$$

is nonnegative. Subscript calculations are always carried out modulo M. Using (7.28), we rewrite (7.34) as

$$-\sum_l d_l^T \cdot C_{l,l+1} \cdot d_l - \sum_l d_l^T \cdot C_{l,l-1} \cdot d_l + \sum_l d_l^T \cdot C_{l,l+1} \cdot d_{l+1} + \sum_l d_{l+1}^T \cdot C_{l,l+1} \cdot d_l. \tag{7.35}$$

Using (7.29), we rewrite this expression as

$$\sum_{l \neq j} \left[d_l^T \cdot C_{l,j} \cdot d_j - d_l^T \cdot C_{l,j} \cdot d_l \right]. \tag{7.36}$$

In (7.36), we reverse the roles of i and j and take the average of the resulting expression and (7.36), obtaining

$$-\frac{1}{2} \sum_{l \neq j} (d_l - d_j)^T C_{l,j} (d_l - d_j) \ge 0. \tag{7.37}$$

This completes the proof of proposition 7.1. ∎

As a very simple special case, we consider an immersed circle under tension. That is, the initial configuration of the boundary is given by

$$X_0(s) = \begin{pmatrix} A \cos \frac{s}{a} \\ A \sin \frac{s}{a} \end{pmatrix}, \tag{7.38}$$

where $A>a>0$ and $s\in[0;2\pi a]$. We show that a solution of (7.7) is given by

$$\mathbf{u} = \mathbf{0} \tag{7.39}$$

$$p = \begin{cases} S(\frac{1}{a}-\frac{1}{A}) & \text{if } \|\mathbf{x}\|\le A \\ 0 & \text{otherwise} \end{cases} \tag{7.40}$$

$$\mathbf{X}(s,t) = \mathbf{X}_0(s) \tag{7.41}$$

At time 0, the tension T is

$$T = S(\frac{A}{a}-1). \tag{7.42}$$

Since

$$\boldsymbol{\tau} = \begin{pmatrix} -\sin(\frac{s}{a}) \\ \cos(\frac{s}{a}) \end{pmatrix}, \tag{7.43}$$

the distribution $\mathbf{F}$ at time 0 is given by

$$(\mathbf{F},\mathbf{w}) = \int_0^{2\pi a} S(\frac{A}{a}-1)\frac{1}{a}(-\cos(\frac{s}{a}),-\sin(\frac{s}{a})\cdot\mathbf{w}(A\cos\frac{s}{a},A\sin\frac{s}{a})ds. \tag{7.44}$$

We show that this is the distributional gradient of the function p defined by (7.40). By definition,

$$(\nabla p,\mathbf{w}) := -(p,\nabla\cdot\mathbf{w}). \tag{7.45}$$

Using (7.40) and the divergence theorem, we obtain

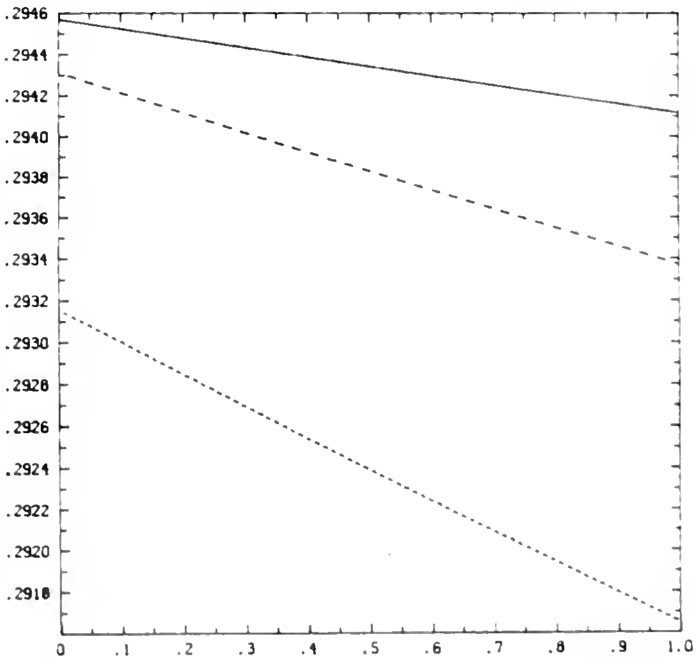
$$(\nabla p,\mathbf{w}) = -\int_{\|\mathbf{x}\|\le A} S(\frac{1}{a}-\frac{1}{A})\nabla\cdot\mathbf{w}(\mathbf{x})d\mathbf{x} \tag{7.46}$$

$$(\nabla p,\mathbf{w}) = -S(\frac{1}{a}-\frac{1}{A})\int_{\|\mathbf{x}\|=A} \mathbf{w}\cdot\mathbf{n}ds. \tag{7.47}$$

(7.47) is equal to (7.44).

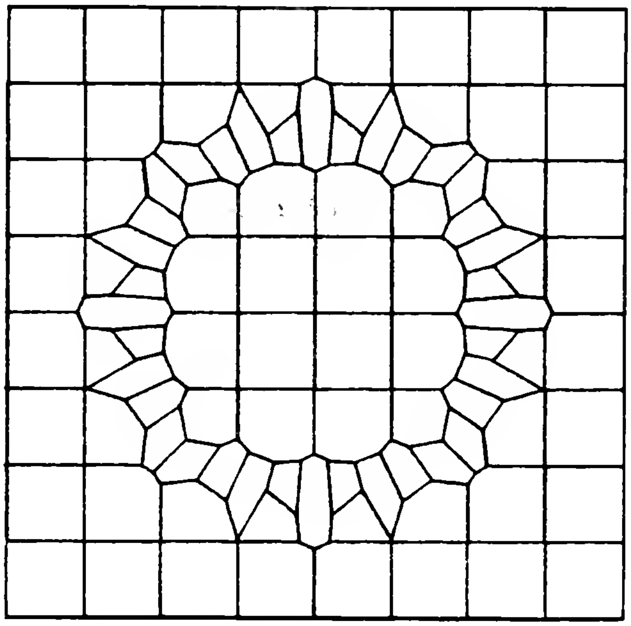
In summary, an immersed circle under tension rests in the fluid, it is prevented from shrinking by an increased pressure in the interior of the circle and the pressure difference is $S(\frac{1}{a}-\frac{1}{A})$.

The circle does not shrink because of the incompressibility. Therefore, this problem provides an opportunity to test how well the incompressibility condition is approximated by the numerical method. If the circle shrinks in the numerical computations, there are two possible explanations. Either markers which are inside the circle at time 0 leave the circle - there is nothing in the method that explicitly prevents this -, or the markers in the circle are more densely packed at later times than at time 0, in violation of the incompressibility condition. We have found that shrinking never occurs for the first reason. Markers which are inside the circle at time zero remain inside the circle. Shrinking does occur for the second reason. The following figure shows the results of a computation in which the initial configuration was a circle around (0.5,0.5) with a radius 0.30625. Δs is chosen such that the circle with radius 0.25 is a resting configuration of the boundary. The curves show the area included by the computed boundary plotted against time for different numbers N of markers. The lowest curve corresponds to N=100,M=36, the middle one to N=400,M=76, the upper one to N=1600,M=156. $\Delta t = \frac{h_N}{2}$, $\nu = 1.0$ and S=0.5. We use the modified fractional step method (6.1b) with two iterations of the projection algorithm in each step. All scalar Helmholtz and Poisson problems are solved up to rounding errors; see the remark below concerning this point.



The amounts of area lost at time 1 are 0.00150 for N=100, 0.00094 for N=400 and 0.00045 for N=1600. Thus the loss of area appears to decrease for increasing N.

It is crucial to use more than one iteration for each Helmholtz or Poisson problem, since the mesh is quite degenerate along the immersed boundary. We illustrate this by the following figure, showing the Voronoi diagram at time 0 for N=100. This degeneracy causes some problems for our two-level algorithm because it uses a uniform auxiliary grid. One might, of course, consider using auxiliary grids which also have a higher concentration of points along the immersed boundary than elsewhere. We have not yet tested this possibility.



For large S, we have observed unstable behavior. After shrinking slowly for a few time steps, the circle suddenly breaks, rapidly shrinking down to the resting radius of 0.25.

8. Open problems and plans for future work

The weakest piece in the method which we have studied is probably the projection algorithm 5.1. The fact that the algorithm, when run over several iterations, sometimes requires a large power k in the smoothing factor

$$(I + \omega L)^k,$$

indicates a stability problem which is not satisfactorily removed by the smoothing introduced in section 5. At the present stage of this work, the smoothing step is useful mainly as a guarantee that nothing will go wrong within the first and second iteration of the projection algorithm. It does not really make it feasible to run the algorithm to convergence. To overcome this difficulty, the definitions of the operators D and $\underline{G}$ should perhaps be modified. The experiment with D_2 presented in section 5, and some similar experiments about which we have not reported here, indicate, however, that slight changes in the definitions of D and $\underline{G}$ may lead to a sharp decrease in accuracy in the projection step, possibly even to a complete loss of pointwise convergence. For a better understanding of this observation, a proof of the convergence shown in table 5.1 may be crucial.

In all the experiments mentioned in sections 6 and 7, we have taken Δt to be δh_N, where δ was a constant not much smaller than 1. We have observed cases in which reduction of δ *increases* the error, in a way similar to the increase in error observed when decreasing the viscosity coefficient ν. This may have the following explanation. The projection step may amplify certain high frequency errors which do not always destroy the accuracy because they are sufficiently damped by the smoothing provided by the viscosity. In the fractional step method (6.1), this smoothing appears in the form

$$\tilde{\underline{u}}^{n+1} := (I-\omega L^n)^{-1}(\underline{u}^n+\Delta t\underline{f}^{n+1}),$$

where $\omega=\nu\Delta t$. If N is fixed and ω is decreased, the smoothing step becomes less effective and may fail to damp the high frequency errors introduced by the projection step sufficiently.

The accuracy of the method should be increased. One can easily and in various different ways obtain $O(h^3)$ in (3.41) for the divergence and gradient operators, thus obtaining pointwise consistent discretizations. It is, of course, not certain that such operators are superior to the weakly first order consistent operators $D,\underline{G}$ which we have used. We have tested one such possibility. It turned out to give worse results than $D,\underline{G}$ when used within the projection algorithm in section 5. Pointwise consistent discretizations of more than first order would probably require larger stencils.

The multigrid method described in section 4 has the disadvantage of not being vectorizable. It may therefore be useful to replace the relaxation scheme by an analogous scheme using several colors. On the torus, the minimum possible number of colors is 7; see Hilbert and Cohn-Vossen (1932). The proof that 7 colors suffice easily leads to an efficient algorithm for the computation of such colorings. It may, however, be sufficient to use a partitioning into less than 7 colors so that the coupling in the discrete Helmholtz operator between points belonging to one color is not necessarily zero but relatively weak. A relaxation sweep through one such color could then be carried out Jacobi-like, and could be completely parallelized. A similar algorithm has been used successfully to solve certain difference equations on regular grids, using a compact 9-point stencil; see Stüben and Trottenberg (1981), p. 113. In this algorithm, only two different colors are used, with weak coupling within each color.

We also plan further work on the algorithm discussed in section 7, an implementation of the method with the boundary condition

$$\underline{u} = 0 \quad \text{on} \quad \partial\Omega$$

and an implementation in three space dimensions.

Bibliography

Aggarwal, A., B. Chazelle, L. Guibas, C. O'Dúnlaing and C. Yap, Parallel Computational Geometry, preprint (1985)

Augenbaum, J.M., A New Lagrangian Method for the Shallow Water Equations, Ph.D. thesis, New York University (1982)

Augenbaum, J.M. and C.S. Peskin, On the Construction of the Voronoi Mesh on a Sphere, J. Comput. Phys. 59, 177 (1985)

Börgers, C. and C.S. Peskin, A Lagrangian Method Based on the Voronoi Diagram for the Incompressible Navier-Stokes Equations on a Periodic Domain, *Proceedings of Free-Lagrangian Methods Conference*, Springer-Verlag, to appear (1985)

Bowyer, A., Computing Dirichlet Tessellations, Computer J. 24, 162 (1981)

Boris, J.P., A Vectorized "Nearest-Neighbors" Algorithm of Order N Using a Monotonic Logical Grid, NRL Memorandum Report 5570 (1985)

Brandt, A., Guide to Multigrid Development, *Multigrid Methods*, Lecture Notes in Mathematics 961, Springer-Verlag (1981)

Brandt, A. and N. Dinar, Multigrid Solutions to Elliptic Flow Problems, *Numerical Methods for Partial Differential Equations*, S.V. Parter, ed., 53-147, Academic Press (1979)

Chan, R.K.-C., A Generalized Arbitrary Lagrangian-Eulerian Method for Incompressible Flows with Sharp Interfaces, J. Comput. Phys. 17, 311 (1975)

Chorin, A.J., Numerical Solution of the Navier-Stokes Equations, Math. Comp. 22, 745-762 (1968)

Chorin, A.J., On the Convergence of Discrete Approximations to the Navier-Stokes Equations, Math. Comp. 23, 341-353 (1969)

Ciarlet, P.G., *The Finite Element Method for Elliptic Problems*, North-Holland, 1978

Crowley, W.P., FLAG: A Free-Lagrange Method for Numerically Simulating Hydrodynamic Flows in Two Dimensions, *Proc. Second International Conference on Numerical Methods in Fluid Dynamics*, Springer-Verlag (1971)

Dukowicz, J., Lagrangian Fluid Dynamics Using the Voronoi-Delaunay Mesh, *Numerical Methods for Coupled Problems*, Pineridge Press, Swansea, U.K. (1981)

Fritts, M.J., Two-Dimensional Lagrangian Flui Dynamics Using Triangular Grids, *Finite-Difference Techniques for Vectorized Fluid Dynamics Calculations*, D.L. Book, ed., 98-116, Springer-Verlag (1981)

Fritts, M.J., Three-Dimensional Algorithms for Grid Restructuring in Free-Lagrangian Calculations, presentation at *Free-Lagrangian Methods Conference* (proceedings to be published by Springer-Verlag), Hilton Head Island, South Carolina (1985)

Fritts, M.J. and J.P. Boris, The Lagrangian Solution of Transient Problems in Hydrodynamics Using a Triangular Mesh, J. Comput. Phys. 31, 173 (1979)

Harlow, F.H. and J.F. Welch, Numerical Calculation of Time Dependent Viscous Incompressible Flow of Fluid with a Free Surface, Phys. Fluids 8, 2182-2189 (1965)

Hilbert, D. and S. Cohn-Vossen, *Anschauliche Geometrie* (1932), English translation *Geometry and the Imagination*, Chelsea Publishing Company, New York (1952)

Hirt, C.W., A.A. Amsden and J.L. Cook, An Arbitrary Lagrangian-Eulerian Computing Method for all Flow Speeds, J. Comput. Phys. 14, 227-253 (1974)

Löhner, R., K. Morgan, J. Peraire and O.C. Zienkiewicz, Recent Developments in FEM-CFD, presentation by R. Löhner at *Free-Lagrangian Methods Conference* (proceedings to be published by Springer-Verlag), Hilton Head Island, South Carolina (1985)

MacNeal, R.H., An Asymmetrical Finite Difference Network, Quarterly of Appl. Math. 11 (1953)

Peskin, C.S., A Lagrangian Method for the Navier-Stokes Equations with Large Deformations, unpublished manuscript (1979)

Peskin, C.S., Numerical Analysis of Blood Flow in the Heart, J. Comput. Phys. 25, 220-252 (1977)

Peskin, C.S. and D. McQueen, Modeling Prosthetic Heart Valves for Numerical Analysis of Blood Flow in the Heart, J. Comput. Phys. 37, 113-132 (1980)

Shamos, M.I. and D. Hoey, *Proc. 16th Annual IEEE Symposium on Foundations of Computer Science*, Berkeley (1975)

Sibson, R., Locally Equiangular Triangulations, Computer J. 21, 243 (1978)

Stetter, H.J., The Defect Correction Principle and Discretization Methods, Numer. Math. 29, 425-443 (1978)

Stüben, K., Algebraic Multigrid (AMG): Experiences and Comparisons, Appl. Math. Comp. 13, 419 (1983)

Stüben, K. and U. Trottenberg, Multigrid Methods: Fundamental Algorithms, Model Problem Analysis and Applications, *Multigrid Methods*, Lecture Notes in Mathematics 961, Springer-Verlag (1981)

Trease, H.E., A Two Dimensional Free Lagrangian Hydrodynamics Model, Ph.D. Thesis, University of Illinois at Urbana-Champaign (1981)

Voronoi, G., Recherches sur les parallélloèdres primitifs, J. Reine Angew. Math. 134, 198 (1908)

NYU COMPSCI TR-183 c.2
Borgers, Christoph
A Lagrangian fractional
step method...